METHODS FOR IMPROVING THE TRACTABILITY OF THE BLOCK SEQUENCING PROBLEM FOR OPEN PIT MINING

by

Martin P. Gaupp

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 JUL 2008	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Methods For Improving The Tractability Of The Block Sequencing Problem For Open Pit Mining	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Colorado School of Mines**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **AFIT/ENEL**	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **Cl09-0017**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**158**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mineral Economics).

Golden, Colorado

Date _____

Signed: _____
Martin P. Gaupp

Approved: _____
Dr. Alexandra M. Newman
Associate Professor
Thesis Advisor

Golden, Colorado

Date _____

_____
Dr. Roderick Eggert
Professor and Director,
Division of Economics and Business

ABSTRACT

A surface mine optimizes its profits by maximizing the net present value (NPV) of minerals extracted from the orebody. This is accomplished by creating a production schedule that defines when each section, or block, of ore is removed. Doing so efficiently requires adherence to geospatial and operational constraints. A common exact method for determining this block extraction sequence is formulating the problem as a mixed integer program where each block is a time-indexed binary variable representing when (and if) a given block is removed from the orebody. We describe the complexities involved in such a formulation and suggest methodologies to expedite the solution times for instances of this block sequencing problem. We adopt three approaches to make the model more tractable: 1) we apply deterministic variable reduction techniques to eliminate blocks from consideration in the model; 2) we produce cuts that strengthen the model's formulation; and 3) we employ Lagrangian relaxation techniques. These three techniques allow us to determine an optimal (or near-optimal) solution more quickly than solving the monolith (original problem). Applying our techniques to data sets ranging from 100 to 10,000 blocks reduces solution times by over 90%, on average.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES vi

LIST OF TABLES vii

ACKNOWLEDGMENTS viii

Chapter 1 INTRODUCTION 1

1.1 Open Pit Mining 2
1.2 Ultimate Pit-Limits Based Approaches to Block Sequencing 5
1.3 Comprehensive Approaches to Block Sequencing 7
1.4 Research Objectives and Expected Outputs 8

Chapter 2 LITERATURE REVIEW 10

2.1 Determining the Ultimate Pit Limits 10
2.2 Ultimate Pit Limits-Based Approaches to Block Sequencing 12
2.3 Comprehensive Approaches to Block Sequencing 16

Chapter 3 THE MODEL 21

3.1 Defining the Block Sequencing Problem 21
3.2 Conceptual Framework 23
3.2.1 Two-Dimensional Conceptual Models 25
3.2.2 Three-Dimensional Conceptual Models 27
3.3 Mathematical Formulation 29
3.3.1 *At* Time t Formulation 32
3.3.2 *By* Time t Formulation 34

Chapter 4 SOLUTION METHODOLOGIES 38

4.1 Numerical Example 40
4.2 Earliest Starts and Latest Starts 43
4.2.1 Earliest Starts Based on Maximum Production and Processing Bounds 43
4.2.2 Latest Starts Based on Minimum Production and Processing Bounds 48
4.3 Cut Generation Techniques 53

4.3.1 Cuts in General . . . 53
4.3.2 Two-Way Earliest Starts Cuts . . . 58
4.3.3 Two-Way Latest Starts Cuts . . . 64
4.3.4 Three-Way Earliest Starts Cuts . . . 71
4.3.5 Three-Way Latest Starts Cuts . . . 83
4.3.6 Cuts Involving More than Three Blocks . . . 96
4.3.7 Using the *by* vs *at* Formulation in Cut Generation . . . 96
4.4 Lagrangian Relaxation Methods . . . 98
4.4.1 Basic Idea Behind the Lagrangian Relaxation Method . . . 100
4.4.2 Implementation of the Lagrangian Relaxation Method for our Problem . . . 105
4.4.3 Feasing Routines . . . 111

Chapter 5 NUMERICAL RESULTS . . . 124

5.1 Data . . . 124
5.1.1 Data Pre-processing . . . 125
5.2 CPLEX Parameter Settings . . . 126
5.3 Computational Results . . . 128
5.3.1 Visual Depiction of an Extraction Sequence . . . 128
5.3.2 Computational Results for Earliest Starts, Latest Starts, Cuts, and the Lagrangian Relaxation Procedure . . . 129
5.3.3 Comparison of Results with Commercial Software . . . 133

Chapter 6 LIMITATIONS, EXTENSIONS, AND CONCLUSION . . . 136

6.1 Limitations and Extensions . . . 136
6.2 Conclusion . . . 138

REFERENCES . . . 140

APPENDIX A . . . 146

LIST OF FIGURES

3.1 Two-Dimensional Conceptual Models 25

3.2 3-D Plus Sign Upper Level Block Example 28

3.3 3-D Rectangular Upper Level Block Example 28

4.1 Two-Dimensional Numerical Example 41

4.2 Earliest Starts Numerical Example 47

4.3 Latest Starts Numerical Example 52

4.4 Two-Way Earliest Starts Cuts Numerical Example 63

4.5 Two-Way Latest Starts Cuts Numerical Example 69

4.6 Three-Way Earliest Starts Cuts Numerical Example 81

4.7 Three-Way Latest Starts Cuts Numerical Example 94

4.8 Feasing Routine for Maximum Constraints Numerical Example 117

4.9 Feasing Routine for Minimum Constraints Numerical Example 122

5.1 Visual Depiction of Micro Pit Results 129

LIST OF TABLES

5.1 Data Sets Used to Empirically Test our Methodologies 125

5.2 Summary of Generation Times for Predecessor Lists, Earliest Starts, Holder Lists, Latest Starts, and Cuts 130

5.3 Summary of Results from Implementing Earliest Starts, Latest Starts, Cuts, and the Lagrangian Relaxation Procedure 132

5.4 MSEP's Results for the 1,060, 1,980, 2,880, and 10,819 Data Set Instances 134

A.1 Detailed Results for the *1,060* Data Set 146

A.2 Detailed Results for the *1,980* Data Set 147

A.3 Detailed Results for the *2,880* Data Set 147

A.4 Detailed Results for the *10,819* Data Set 147

A.5 Detailed Results for the *10,819A* Data Set 148

A.6 Detailed Results for the *10,819B* Data Set 148

A.7 Detailed Results for the *10,819C* Data Set 148

A.8 Detailed Results for the *10,819D* Data Set 149

A.9 Detailed Results for the *10,819E* Data Set 149

A.10 Detailed Results for the *10,819F* Data Set 149

A.11 Detailed Results for the *10,819G* Data Set 150

A.12 Detailed Results for the *Newmont* Data Set 150

ACKNOWLEDGMENTS

First and foremost I thank my Lord and Savior Jesus Christ for giving me the ability and perseverance to complete my PhD. I am continuously amazed at how His hand is involved in every aspect of my life.

I wish to express deepest thanks to my thesis adviser Dr. Alexandra Newman at the Colorado School of Mines for her guidance in formulating the model examined in this paper and her vigilance in proof-reading and reviewing all editions of the written work. I would also like to thank my dissertation committee comprised of Dr. Michael Heeley, Dr. Daniel Kaffine, Dr. Enrique Rubio, and Dr. Luis Tenorio for their assistance in making this endeavor a success. Special thanks to fellow PhD student Hendro Fujiono for analyzing my data in MineSight Economic Planner. I also wish to thank the U.S. Air Force for sponsoring me for this PhD over the past three years.

Lastly, I wish to thank the three most important ladies in my life, my wife Britta, my daughter Amanda, and my daughter-to-be Keira for all their love and steadfast patience throughout the past three years. I could not have done it without them!

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Chapter 1

INTRODUCTION

Mining is a complex, expensive, but potentially lucrative business. Today's open pit mines are huge projects that make extensive use of automated equipment and employ the latest technology to ensure a profitable enterprise. The tight profit margins under which these mines often operate and the finicky nature of mineral markets require efficient ore removal schemes to ensure that the mine makes money.

Scheduling the operations at an open pit mine involves determining what material to remove at which time. The material in the mine is divided into rectangular blocks whose size is based on the operation's production and processing capacities. To maximize the net present value of all the mineral in the orebody, a mine engineer must create an optimal extraction schedule. The extraction schedule must obey all geospatial sloping rules so the resulting hole does not cave in on itself, as well as operational constraints that limit the amount of material produced at the site and the amount of material processed at the mill. The ensuing schedule represents the optimal block extraction sequence, and can potentially produce millions of dollars or more in revenue compared to ad hoc extraction schemes. Determining this optimal extraction policy, however, is a computationally intensive undertaking. Attempts to reduce solve times via heuristics often fail to give an optimal solution.

Although quick to provide a solution, heuristics have no guarantee of optimality and may not give any indication of the correctness of the solution. On the other hand, spending countless hours or even days to derive the optimal solution is costly and cumbersome as it gives mine planners little ability to quickly react to changing conditions at the mine or in the marketplace by using the model. We endeavor

to arrive at an optimal solution by using deterministic data reduction techniques to decrease the number of variables in the problem and by using stronger problem formulations that tighten the feasible region via cuts. Also, we employ Lagrangian relaxation techniques in conjunction with heuristics to generate feasible solutions as a means of expediting solution times. Thus, we provide the best of both worlds, optimal solutions in reasonable solve times. Our solution methodologies help mine engineers prudently schedule operations at the mine, allowing for the efficient removal of valuable material from the orebody. Our fast solve times allow mine engineers to update their schedules more frequently than by using other, more time consuming scheduling methods, creating cost-effective schedules more quickly than is currently being done.

1.1 Open Pit Mining

Open pit mining involves extracting material from the Earth's surface downwards; hence, open pit mines are often referred to as *surface mines*. Open pit mines result in crater-like holes in the ground once all the valuable mineral is removed from the orebody.

Orebodies in open pit mines are modeled using a collection of blocks, commonly three-dimensional rectangles. Each block is characterized by a weight and an amount of valuable mineral. The ratio of valuable mineral to total block weight is used to determine whether the block is classified as *ore* or *waste*, which in turn dictates the profit that a block produces. The basic decision a mine planner makes is when, if at all, to remove each block in the orebody of interest. Maximizing the net present value (NPV) of all processed ore at the site ensures that the mine's operations process the most valuable material from the orebody. An efficient extraction schedule coupled with other sound business practices help ensure that the mine's NPV is maximized.

Kriging and other sampling techniques provide information about the extent of the orebody and types of ore on a per block level at the site. Mine planners use this

block information along with geospatial rules governing the order in which blocks can be extracted (known as *sequencing* constraints) as well as on site *operational* limits to create a block extraction sequence. The mine planner's task is to efficiently guide the extraction of ore from the mine over its entire life span, which frequently is many years long. The optimal block extraction schedule assures that the most valuable ore is removed from the orebody without violating any geospatial or operational constraints.

Differentiating ore from waste is done by establishing a *cutoff grade*. The cutoff grade is heavily affected by the commodity market of the metal being mined; if the metal is worth more, then a lower cutoff grade is used because lower quality ore can still result in profits. The cutoff grade must either be treated as variable in the problem or assumed to be constant. The latter is often the case to ensure a tractable problem formulation.

One set of limitations in open pit mine design is geospatial in nature. Material must be removed from the mine in such a way as to ensure that the hole created does not collapse in on itself. If only a static (i.e., single time period) picture of the mine is required, these *sequencing* constraints are essentially the only constraints that must be adhered to. Maximizing the value of the mineral in the orebody subject to these sequencing constraints results in a final mine layout called the *ultimate pit limits* (note, the ultimate pit limits assume a fixed cutoff grade). The ultimate pit limits define an economic envelope of the valuable ore in the orebody and indicate how much the ore is worth in today's dollars (i.e., without accounting for the time value of money).

The solution to the *ultimate pit limits problem* is often used to economically justify a project. If the cost of removing the material to reach the ultimate pit limits is more than the profit of the ore processed from this removed material, then the entire mining venture likely results in negative profits and the project is not advisable. The ultimate pit limits also guide mine planners in locating various on-site features such

as production equipment, maintenance shops, personnel offices, and other support facilities.

Incorporating time into the mine design results in a much more complicated problem—the *block sequencing problem*. Now the mine planner must not only determine *which* blocks to remove, but *when* to actually remove them; hence, *side constraints* must be obeyed. These side constraints enforce requirements on the average grade extracted from the mine per time period and impose lower and upper bounds on the amount of material extracted (i.e., *produced*) and sent to mills (i.e., *processed*) per time period. Average grade constraints, which are only required in formulations with a variable cutoff grade, ensure that *on average* the grade of blocks extracted in a certain time period be between minimum and maximum qualities. Production constraints relate to the amount of equipment the mine has at its disposal per time period, while processing constraints involve per time period limits at the mills to which the removed ore is sent.

Production and processing rates are defined on a *per time period* basis. The actual length of this time period is chosen somewhat arbitrarily. A shorter time period results in a schedule of higher fidelity, but also contains more variables. Generally, a time period is chosen such that an entire block can be extracted and processed completely during the time period; otherwise, blocks could only be partially removed, causing practical implementation issues.

The optimal solution to the block sequencing problem is a time-indexed schedule of when any given block in the orebody should be removed (if it is removed at all) that maximizes the net present value of the ore at the site subject to all sequencing and operational constraints. The sequencing constraints impose slope requirements on the walls of the pit while the operational constraints provide lower and upper bounds on average grade, production rates, and processing rates.

Solving the block sequencing problem poses unique challenges that can be addressed with a diverse set of solution methodologies. Because of the mathematical

structure of the block sequencing problem, its theoretical complexity is exponential. Real-world model instances, like the one studied in this paper, tend to overtax standard mixed integer programming algorithms and make solving the block sequencing problem too cumbersome. Although advances in computer hardware and software have made this problem more tractable, better solution approaches and strong formulations are still required to ensure reasonable computing times for large mines.

There have been two major approaches to solving the block sequencing problem. The first, which we call the *ultimate pit limits-based approach*, decomposes the problem into three stages and sequentially solves each stage of the problem to arrive at a mine schedule. The second, known as the *comprehensive approach*, takes a global view of the problem to determine the optimal block extraction sequence, thus avoiding sub-optimal mine schedules that are inherently created using the former *ultimate pit limits-based approach.*

1.2 Ultimate Pit-Limits Based Approaches to Block Sequencing

Traditionally, the ultimate pit limits are calculated first and then a series of pushbacks is determined to create a schedule of operations for the mine. One generates a series of nested pits by gradually increasing the ore price of the material being mined. The mine planner chooses a fictitious ore price and determines which blocks are mined and which are left unextracted. This results in a set of blocks that defines the first nested pit. Next, the mine planner raises the price and determines which of the remaining blocks to extract from the pit. This results in the second nested pit. The process continues until all blocks in the orebody are analyzed and no further nested pits can be created.

From the series of nested pits, the outermost pit, or the *ultimate pit limits*, is determined. This outermost pit separates the blocks to be mined from those left in the ground. It is important to note that not all blocks that are extracted are profitable in and of themselves. If a block is mined, it contributes to the overall profits of the mine

either by being profitable by itself, or by allowing access to more profitable blocks on lower levels.

Mine planners group neighboring nested pits into pushbacks based on mine production rates, mill processing rates, or other operating constraints that affect the mine. After the pushbacks have been determined, the production sequence is established for blocks in each pushback separately (i.e., only considering blocks in one pushback at a time), and the mine's operations are scheduled over the life of the mine.

This three-stage process of determining the ultimate pit limits, creating nested pits and pushbacks, and then generating a block extraction schedule is the traditional approach to block sequencing. Decomposing the block sequencing problem into these three separate stages forces the mine planner to make certain decisions in each stage without information from the other stages. Making such interrelated decisions independently often sacrifices optimality of the overall block extraction schedule.

The implied assumption that prices increase over time may not be valid, rendering the nested pits sub-optimal. If the nested pits are not optimal, the resultant pushbacks and mine schedule will be sub-optimal as well.

Decomposing the problem into three stages allows certain decisions to be postponed to later stages. Since temporal issues do not enter into the mine planner's decision process until sequencing the blocks within a pushback, orebody-wide block extraction decisions are made without considering the time value of money. When the mine planner is ready to sequence blocks within a given pushback, many blocks have been eliminated from consideration since they fall outside the ultimate pit limits; hence, many blocks are not eligible to be sequenced during that time period. Also, a loose precedence of block extraction has already been established between groups of blocks via their assignment to a given pushback since generally blocks in the first pushback are mined before those in the second, etc.

The three-stage process also does not allow for blocks between different pushbacks to be scheduled for removal in the same time period. In many cases it may be more

profitable to leave some blocks in a previous pushback unmined and start mining a new pushback instead of continuing to mine from a currently-active pushback. By forcing block extraction from just one pushback per time period, the mine planner loses flexibility and potentially creates a sub-optimal mine schedule.

The three-stage process has been necessary due to the vast number of blocks in a typical deposit of interest. Partitioning the problem into three stages makes the problem more manageable. Ideally, an over-arching approach which allows the block sequencing problem to be solved without sacrificing optimality would be preferable. Such a method is available, although solving the resultant problem is significantly more complex.

1.3 Comprehensive Approaches to Block Sequencing

Modern approaches to open pit mine scheduling abandon the three-stage process in the traditional approach for a process that simultaneously determines the ultimate pit limits, pushbacks, and block extraction sequence. Each block in the orebody is analyzed and assigned a time period at which it should start being mined (if at all). From this block extraction sequence, the mine planner can create pushbacks based on operational constraints at the mine (such as production and processing rates). The pit that results once mine operations are complete (i.e., all blocks scheduled for extraction have been removed) represents the ultimate pit limits which portray the optimal solution with respect to the net present value of all available ore at the site. Hence, modern approaches replace the traditional, sub-optimal, three-stage process with an optimal, comprehensive, one-stage approach.

Since the ultimate pit limits are not necessary to create the pit's schedule, comprehensive approaches to open pit mine scheduling do not explicitly calculate the ultimate pit limits. In actuality, the static ultimate pit limits might be significantly different than the final pit outline achieved once the block sequencing problem has been solved using the comprehensive approach. This is because the time value of

money and other operational constraints are incorporated into the block sequencing problem but completely ignored by the ultimate pit limits problem.

Although the comprehensive approach to block sequencing results in a problem formulation that is less tractable and harder to solve, the solution gained is a truly optimal mine schedule that considers all blocks in the orebody, not just those in an arbitrary pushback. The resultant optimal block extraction sequence implicitly results in a schedule of pushbacks and the ultimate pit limits, all incorporating a discount factor to account for the time value of money.

1.4 Research Objectives and Expected Outputs

The objective of this research is to increase tractability for a variant of the block sequencing problem. This problem is commonly formulated as a mixed integer program (MIP), which is then solved via the branch-and-bound algorithm. Heuristics are also frequently used, but heuristic solution methods often do not provide a means for assessing the degree of optimality achieved; they fail to bound the problem's objective function value. However, as the number of decision variables (in our case, the number of blocks in the mine) increases, computation times for the MIPs increase exponentially. For even small, real-world applications with thousands of variables, this leads to solution times of days or weeks, which is far too slow for commercial applications. As a result, most commercial mine planning software (e.g., Whittle, Vulcan, or MineSight Economic Planner) uses heuristics to create a mine schedule, albeit the methodology these commercial packages use is protected as a trade secret.

We also formulate our problem as a mixed integer programming problem and use the branch-and-bound algorithm to solve it. We examine the differences in tractability between equivalent formulations of the model based on decision variable definitions. We investigate various deterministic and heuristic solution techniques available for solving mixed integer programming problems. Among these are variable elimination techniques, formulation strengthening via cut generation, and decomposition-based

relaxation approaches to solving mixed integer problems. Our hope is that one (or a combination) of these techniques significantly reduces computation times.

We apply various deterministic data reduction techniques to: 1) eliminate blocks from consideration in the model due to their inability to be reached before a certain time period (i.e., defining *earliest start times*); 2) pre-set the mining decision of certain blocks in the model due to their position in the mine (i.e., setting *latest start times*). These techniques reduce the number of decision variables (and implicitly the number of constraints) that must be examined by the algorithm, thus strengthening the problem's formulation and expediting solution time.

We attempt to produce cuts that strengthen the model's formulation, making the problem more tractable. Although the generation of these cuts requires a significant amount of time, the reduction in the original problem's solution time that they afford makes the initial investment worthwhile.

Lastly, we investigate Lagrangian relaxation techniques that allow us to converge to the optimal solution more quickly than solving the monolith (original problem).

Reducing the solution time for the block sequencing problem allows mine planners to efficiently schedule larger orebodies that until now were too big to handle. Mine planners now have at their disposal a tool that can give them an optimal extraction schedule rather than a best guess that is sub-optimal and potentially misleading (i.e., a mine schedule for a mine which should never have been considered for extraction).

Chapter 2

LITERATURE REVIEW

2.1 Determining the Ultimate Pit Limits

The ultimate pit limits represent the set of blocks in the pit that maximizes the total profit of the pit based on each block's current net profit and its physical location in the mine. A block's net profit is the difference between the total value of the extracted mineral in the block and the cost of extracting that material from the mine and processing the block at a mill. A block's physical location is important because blocks on lower levels of the mine cannot be accessed until those blocks above it are removed. The ultimate pit limits represent a static boundary of blocks that maximizes the non-discounted value of the mineral in the orebody.

Solving the ultimate pit limits problem is extensively reviewed in the literature. Laurich (1990) and Thomas (1996) provide excellent overviews of the literature applicable to this problem. Kim (1978) and Hulse (1992) address various ways that a block can be defined, an issue that must be resolved in order to determine the layout of the ultimate pit. Achireko and Frimpong (1996) use neural networks to examine the stochastic nature of a block's grade and weight characteristics, showing that the assumed homogeneity of a block's grade or weight could severely alter the ultimate pit.

Dynamic programming methods use intelligent enumeration to determine the ultimate pit limits. Lerchs and Grossman (1965) use dynamic programming methods to solve two-dimensional versions of the ultimate pit limits problem. Koenigsberg (1982), Wilke and Wright (1984), Shenggui and Starfield (1985), and Wright (1987) also use dynamic programming to solve the ultimate pit limits problem, but tackle

the three-dimensional version.

Another method of solving the ultimate pit limits problem is by using network flow algorithms. Network flow formulations model the orebody as a network of nodes representing the blocks in the mine connected by arcs representing the sequencing constraints between the blocks. The goal of the model is to maximize the flow from a source node to a sink node. When the algorithm terminates, those arcs that originate at the source and have excess capacity represent profitable blocks that should be mined in the optimal solution. These profitable blocks and any other blocks that must be mined to reach them represent the ultimate pit limits. In the same paper in which they use dynamic programming to solve two-dimensional versions of the ultimate pit limits problem, Lerchs and Grossman (1965) model an open pit mine as a weighted, directed graph where vertices represent blocks and arcs represent mining restrictions (i.e., sequencing constraints). They determine the ultimate pit limits by solving for the maximum closure of this graph. Zhao and Kim (1992) use a similar approach with some modifications that they claim solve problem instances faster than the Lerchs and Grossman method. Johnson (1969), Picard (1976), Yegulalp et al. (1993), Hochbaum and Chen (2000), and Hochbaum (2001) also use network flow algorithms to derive solutions for the ultimate pit limits problem.

Heuristics use short-cuts to solve the ultimate pit limits problem; however, many are either flawed or not capable of bounding the objective function values they derive. Pana (1965) creates the floating cone heuristic, also known as the moving cone or dynamic cone heuristic. Laurich (1990) and Thomas (1996) review the general idea behind Pana's original algorithm. Korobov (1974) attempts to improve on Pana's method. Dowd and Onur (1992) attempt to correct the flaws in Korobov's algorithm. Laurich (1990) reviews a constructive heuristic known as the incremental pit expansion method. Tamatomi et al. (1995) outline the latest developments in floating cone methods. Alford (1995) creates a new version of the floating cone algorithm called the floating stope method, but it too results in sub-optimal ultimate pit limits.

Exact optimization approaches are also common means of solving the ultimate pit limits problem. Gershon (1982), Cai (1989), and Huttagosol and Cameron (1992) use linear programming methods to solve the ultimate pit limits problem. Ovanic and Young (1995) use a branch-and-bound approach incorporating special ordered sets to solve the ultimate pit limits problem.

2.2 Ultimate Pit Limits-Based Approaches to Block Sequencing

The ultimate pit limits-based approach to open pit mine scheduling was the original method used to solve the block sequencing problem (see Section 1.2 for more explanation). This traditional approach served as a bridge between the ultimate pit limits problem and the block sequencing problem, and an extensive amount of literature appears on the topic.

Osanloo, Gholamnejad, and Karimi (2007) provide an excellent overview of the models and algorithms used to solve the long-term open pit mine production planning problem. Their review includes both deterministic and stochastic approaches. With respect to deterministic approaches, they address solely ultimate pit limits-based approaches.

Johnson (1968) uses Dantzig-Wolfe decomposition principles to solve the block sequencing problem. He breaks the multi-time period problem into single-time period problems which he solves as ultimate pit limit problems. The master problem enforces operational constraints (i.e., production and processing requirements) while the sub-problems enforce sequencing constraints. According to Osanloo, Gholamnejad, and Karimi (2007), his methodology results in sequencing constraint violations and can only handle a relatively small data set.

Osanloo, Gholamnejad, and Karimi (2007) show how Roman (1974) uses dynamic programming to enumerate block sequences and then select the optimum sequence based on maximizing NPV. This method becomes far too cumbersome with even small data sets, and there is no guarantee that production and processing con-

straints are adhered to in each time period.

Gershon (1982) describes a linear programming application that optimizes the scheduling of mining operations. His block scheduling optimization concept uses a mathematical model to determine the most profitable mine operation scheme from pit-to-plant-to-market. By accounting for pit-plant-market interfaces and optimizing operations over the entire life of the mine, Gershon's model is able to calculate long, intermediate, and short-range mine plans. His mine scheduling optimization concept determines the ultimate pit limits for a specific time period subject to equipment availability, plant requirements, and market conditions, intrinsically creating an extraction schedule which eventually results in the ultimate pit limits. Because his method is based on linear programming, it allows for partial blocks to be mined if all precedent blocks have been completely removed.

Dagdelen (1985) uses Lagrangian relaxation to solve the block sequencing problem. His methodology resorts to elastic operational constraints as a means of overcoming the problem of infeasibility that results when using optimal decision variable values from the solution to the Lagrangian relaxation subproblem in the monolith.

Gershon (1987) develops a mine scheduling heuristic based on a block's positional weight, "the sum of the ore qualities within the cone generated downward from a block within the ultimate pit," (Gershon, 1987, p. 8) to determine when a block should be mined. The positional weight of a block defines the desirability of removing that block at a particular point in time; higher positional weights are more desirable. The accessible block with the highest rank is extracted and then the entire process, starting from determining the positional weight of the remaining blocks in the ultimate pit, is conducted again until all the blocks in the ultimate pit have been removed. Wang and Sevim (1992) use Gershon's heuristic without requiring prior knowledge of the actual outline of the ultimate pit.

Thomas (1996) reviews a process called the nested Lerchs-Grossman algorithm that uses the Lerchs-Grossman algorithm to create a series of nested pits which is

aggregated into pushbacks. A mine planner then schedules the block extraction sequence for each individual pushback, creating an overall mine schedule. The nested Lerchs-Grossman algorithm is based on parametric analysis in which the development of the pit is characterized by gradual modification of one or more key parameters. The parameter chosen by Lerchs and Grossman is the amount by which the economic value of each block in the model is reduced. As this economic value of each block is progressively increased past certain critical values, the ultimate pit limits contour changes to enclose a smaller volume. The end result is a series of nested pits. Therefore, this technique is referred to as the Nested Lerchs-Grossman algorithm. However, gaps between nested pits can result in infeasible schedules where production or processing capacity cannot reconcile the difference between consecutive nested pits, thus essentially violating the block extraction sequencing constraints.

Dowd and Onur (1992) use dynamic programming to schedule the extraction sequence of blocks in the ultimate pit. Noting that their method suffers from exponential growth in processing time as the number of blocks increases, they point out that many block extraction sequences can be eliminated because they are infeasible, thus reducing the search space. They concede, however, that eliminating infeasible block sequences only helps the problem incrementally and that dynamic programming is still impractical for large mines.

Tolwinski and Underwood (1992) use dynamic programming to determine the optimal production schedule for an open pit mine that satisfies both physical and economic constraints. They model the evolution of the mine as a sequential optimization problem by generating a sequence of pits which starts at an initial pit or unmined surface and proceeds through incrementally larger pits until the ultimate pit is created.

Onur and Dowd (1993) point out that the extraction schedule and the physical means of extracting the ore (via haul roads and safety berms) must be incorporated into the final mine plan. They describe a simple block construction that allows haul

roads to be incorporated into the pit's design.

Wang and Sevim (1995) present an alternative to the economic block value parameterization method used by Lerchs and Grossman for finding a series of nested pits for production planning. They point out that the method of parameterization which is commonly used to create a series of nested pits often results in consecutive pits that are too far apart from each other because operational constraints do not allow the leap from one nested pit to its neighbor. Wang and Sevim create an algorithm that can generate nested pits with controllable size increments (i.e., controllable gaps) to meet mine-specific production and/or processing capacity limits, thus overcoming the gap problem.

According to Osanloo, Gholamnejad, and Karimi (2007), Tolwinski and Golosinski (1995) and Tolwinski (1998) propose a method of solving the block sequencing problem based on a depth-first search technique. Although their methodology handles operational constraints well and is usable for large data sets, it does not guarantee optimum results with respect to NPV maximization.

Halatchev (2002) emphasizes the difference in time and spatial planning aspects of an open pit mine. He shows that while the spatial aspect addresses the determination of the shape and size of pit benches, the time aspect treats the sequencing of mine extraction activities. In doing so, he describes the interdependencies of these two ideas and clearly shows the short comings of the ultimate pit limits-based approaches to block sequencing.

Dagdelen (2005) shows how the block sequencing problem involves the determination of a cutoff grade, which in turn is used to decide whether or not a given block should be mined and when it should be mined, and then lastly, once mined, how the ore should be processed. He describes the circular relationship between the physical capacities of the mining operations (which, taken with production costs, determine the ultimate pit) and the design of pushbacks (which, based on the cutoff grade, are used to actually schedule the extraction of blocks).

Ramazan (2001) and (2007) uses a clustering idea to classify the block data into similar entities (which he calls *fundamental trees*). Fundamental trees aggregate blocks of material, thereby decreasing the number of integer variables and the number of constraints required within the MIP formulation. A fundamental tree is any combination of blocks within a given pushback that can be mined with maximum NPV while still obeying sequencing constraints. Unfortunately, since the fundamental trees are only defined within pushbacks, the optimality of this method is completely dependent on the method used to determine the optimal pushback scheme for the ore body.

2.3 Comprehensive Approaches to Block Sequencing

Comprehensive approaches to open pit mine scheduling use a one-stage process to simultaneously determine the ultimate pit limits, pushbacks, and block extraction sequence (see Section 1.3 for more explanation). This approach is the more popular means of attempting to solve the block sequencing problem and recently more literature is appearing on this topic.

Denby and Schofield (1994) use genetic algorithms to simultaneously determine the ultimate pit limits and the orebody's extraction schedule. Using the net present value of the extraction schedule to assess fitness (i.e., the optimality of the schedule), the fittest members of the population reproduce the most, mimicking the survival of the fittest analogy in nature. Crossover (mating) between pit schedules and mutation actions create new generations of pit extraction schedules, ultimately leading to a better (more fit) extraction schedule. Their method develops the ultimate pit limits and the mine's extraction schedule simultaneously, thus avoiding the sub-optimality faced by ultimate pit limits-based approaches. Although optimality may never be reached, they claim their genetic algorithm can find good sub-optimal block extraction schedules.

Osanloo, Gholamnejad, and Karimi (2007) show how Elevli (1995) applies Tol-

winski and Underwood's method to obtain the ultimate pit limits and block extraction sequence simultaneously. However, they point out that Elevli's method suffers from the gap problem and that his method does not provide a mathematically proven optimal solution nor even a feasible solution for large data sets.

Sevim and Lei (1998) describe the simultaneous nature of open pit mine scheduling. They show that long term production planning in open pit mines involves the simultaneous resolution of four issues: 1) the production rate (number of blocks to be mined each year), 2) the specific group of blocks that should be mined in a given year, 3) the cutoff grade to be used to determine ore and waste blocks, and 4) the ultimate pit limits. They depict these four issues interacting in a circular fashion and propose a process that simultaneously handles all four aspects of the problem. Ultimately, their solution methodology generates a series of nested pits, of which the sequence with the highest NPV is chosen.

Hoerger, Bachmann, Criss, and Shortridge (1999) describe long-term mine extraction and processing scheduling at Newmont's Nevada Operations. Their complex model incorporates over thirty mine sources that feed over sixty possible processing facilities during twenty time periods. Their tool employs linear programming to maximize NPV by matching grade and metallurgical type increments to optimum processing plants or stockpiles subject to production and processing capacity constraints and blending requirements. Additionally, they use integer programming to sequence actual block extractions from various mines and to handle fixed costs.

Lagrangian relaxation is a tactic used to remove complicating side constraints from a mixed integer program and transform the problem into a more tractable formulation. Akaike and Dagdelen (1999) use Lagrangian relaxation to convert their integer-programming formulation into one based on networks. Their underlying problem formulation has a network flow structure with a complicating side constraint in the form of a production capacity constraint. They integrate this production capacity constraint into the objective function, creating a long-term production scheduling

problem with the same characteristics as the final pit design problem. This relaxed problem is then solved via the maximum closure algorithm that Lerchs and Grossman use in their original work with the three-dimensional ultimate pit problem. Akaike and Dagdelen then use an iterative process that alters the values of the Lagrangian multipliers until the solution to the relaxed problem meets the original capacity constraints. Their model does not contain any processing or average grade constraints. Hence, dualizing just the production capacity constraints results in a network structured formulation which is more tractable than their original problem formulation.

Cai (2001) also uses Lagrangian relaxation by incorporating operational constraints into the block's net value calculation via multipliers to penalize violations of these constraints. Cai, however, does point out that using Lagrange multipliers may not result in objective function value convergence to an acceptable solution for all problem instances due to the complexity of the problem and the existence of gaps between nested pits. As such, he admits that it is may be impossible to produce a multi-period schedule using his Lagrangian relaxation methodology.

Erarslan and Çelebi (2001) determine a production schedule to maximize net present value subject to grade, blending, production and other operational constraints. They use dynamic programming to solve their problem for a fixed pit volume. They enumerate various volumes to determine the optimal pit size. In doing so, the authors claim that their method solves the ultimate pit limits problem and the block sequencing problem simultaneously. But, since their procedure is based on a dynamic programming approach, it is not efficient for medium to large data sets.

Kumral and Dowd (2002) use simulated annealing to create an optimal mine operating schedule via a two-stage optimization routine. The first stage uses Lagrangian parameterization that results in an initial sub-optimal solution. This is followed in the second stage by applying multi-objective simulated annealing to further improve the sub-optimal schedule.

Ramazan and Dimitrakopoulos (2004a) present a general description of an effi-

cient mixed integer program for the open pit mine scheduling problem. They aim to maximize the overall discounted net present value of the mine's ore subject to the limitations of wall slope requirements, grade blending requirements, ore production, and mine capacity. They also propose reducing the number of binary variables by separating positively valued blocks (which they call "ore" blocks) from negatively valued blocks (which they call "waste" blocks). Only variables representing ore blocks are defined as binary, while those representing waste blocks remain continuous, thus allowing for partial excavation. Their analysis shows that such a scheme can significantly reduce solution times.

In another paper, Ramazan and Dimitrakopoulos (2004b) show that mixed integer programming formulations fail to produce practical mining schedules due to, *inter alia*, in-situ variability of orebodies. They propose an alternative mixed integer programming formulation that considers the probability of blocks being scheduled in a given production period, thus dealing with the in-situ variability and other practical issues of scheduling patterns. They show how their methodology is especially applicable to poly-metallic deposits, where orebody uncertainty is usually a significant issue.

Menabde, Froyland, Stone, and Yeates (2004) examine the mine optimization problem under uncertainty by using a set of conditionally simulated orebody models. Their formulation simultaneously optimizes the block extraction sequence and the cutoff grade.

Froyland, Menabde, Stone, and Hodson (2004) examine the value of additional information with respect to open pit mining projects. The information used to create a block model is gathered from a series of drillholes that provides a set of data from which a three-dimensional model of the orebody is created. The more drillholes used, the more accurate the data are. However, each drillhole has an associated cost and the information garnered may be of marginal benefit. The authors examine the trade-off between expending additional funds on more information (in the form of

additional drillholes) and the benefits of the information gathered. If the drillhole provides information that significantly alters the mine planner's perception of the orebody, the overall mining operation may become more profitable and make the cost of attaining the additional information worthwhile.

Boland, Fricke, and Froyland (2006) examine the use of knapsack inequalities and covers to create valid and useful cuts to expedite solution time. They show that this method can be used to create cuts for pairs of blocks and larger sets of blocks. However, they contend that in the multiple block case (where more than two blocks are involved) generating the cuts involves a separate optimization problem and the time savings garnered from including the cuts in the mine scheduling problem could easily be lost due to the time required to actually generate the cuts themselves.

Kawahata (2006) expands on the Lagrangian relaxation procedure developed by Dagdelen (1985) and includes a dynamic cutoff grade policy to maximize the pit's NPV in his formulation. He utilizes two Lagrangian relaxation subproblems, one for the most aggressive mine sequencing case and the other for the most conservative mine sequencing case, to bound the optimal solution space. These bounds help eliminate variables from the monolith, thus significantly expediting solution time.

Boland, Dumitrescu, Froyland, and Gleixner (2007) use aggregation techniques to reduce the number of binary variables in their problem formulation. They use aggressive block aggregation to schedule production at the mine (i.e., the removal of material) and then disaggregate their data back into individual blocks to make the processing decisions at the mills. They propose an iterative process to disaggregate the solutions obtained with aggregated data.

Espinoza, Goycoolea, Moreno, and Rubio (2008) create a class of heuristics based on the notion of topological sorting obtained from directed graphs of the resource-constrained open-pit mining problem. Their preliminary results indicate that feasibility pre-processing (an *earliest start* idea) has promise but that post-processing (i.e., *fixing* certain variables) is not very useful.

Chapter 3

THE MODEL

With the goal of maximizing the net present value (NPV) of the ore in the pit, the block sequencing problem endeavors to determine the most efficient extraction sequence of the usable material while meeting sequencing, average grade, production and processing constraints. This NPV is the sum of all the extracted blocks' discounted net profits. A block's profit is calculated as the market value of the ore in the block minus the costs of removing and processing the material in the block, and then discounting this number based on the time of extraction and an appropriate discount factor. The solution to the block sequencing problem is a block extraction schedule indicating when and if each block is scheduled for removal from the pit.

3.1 Defining the Block Sequencing Problem

Assumptions

In order to make our problem more manageable, we make the following assumptions:

- We use a fixed cutoff grade in our model. This means that if a block meets the cutoff grade, then we consider it *ore*; otherwise, we consider it *waste*. Because we use a fixed cutoff grade, the average grade requirements are moot since they are always met. As such, we do not employ average grade constraints in our model formulation.

- We employ a 45° sloping rule when describing our geospatial sequencing requirements. Additionally, we use the plus (+) sign convention when determining

block sequencing in our three-dimensional block model (see Figure 3.2).

- We do not allow fractional mining of blocks. When we start mining a block in a given time period, it is completely removed from the orebody by the end of that time period. The decision to remove a block is an all-or-nothing decision; hence, the decision is easily modeled using binary (0-1) decision variables.

- We do not concern ourselves with any equipment or manpower allocation issues. Our model formulation views the mine from a strategic level and the resultant schedule identifies mine extraction operations on a per-block basis only. Mine planners can then use this block sequencing information to assign resources and labor to actually remove the blocks from the mine.

- Our model is completely deterministic. We assume that we know the location, mineral content, and total material content of each block in the orebody with absolute certainty.

Decision Variables

The decision variables in the block sequencing problem are whether or not to remove any given block during any given time period. A block can be removed at most once. Not all blocks need be removed from the pit. If the orebody consists of $|B|$ blocks and the time horizon for the mine is $|T|$ time periods, then there are $|B| \cdot |T|$ binary decision variables involved in the problem formulation.

Objective Function

The objective of our problem is to maximize the net present value of the ore in the mine. We calculate this NPV by summing all the extracted blocks' discounted net profits. The profit of a block is the market value of the block's ore minus the costs of removing and processing that material. We account for the time value of

money by discounting this profit based on the time period in which we remove the block from the mine.

Constraints

Our objective function is constrained by both geophysical and operational constraints:

- We must obey the sloping requirements and sequencing constraints to ensure that the pit walls do not collapse. As stated in the assumptions above, we use a 45° sloping rule and the plus (+) sign three-dimensional convention.
- Since we assume a fixed cutoff grade, we implicitly adhere to minimum and maximum average grade constraints. As such, these constraints are moot and are not included in our problem formulation for simplicity.
- We must abide by minimum and maximum production constraints for each time period. These constraints bound the amount of material that can be removed from the mine (i.e., *produced* from the orebody).
- We must comply with minimum and maximum processing constraints for each time period. These constraints limit the amount of ore that can be sent to the mills for processing.

3.2 Conceptual Framework

This research applies variable elimination techniques, formulation strengthening methods, and Lagrangian relaxation approaches to improve solution times for the block sequencing problem. We apply the comprehensive approach (discussed in Section 1.3) to solve the block sequencing problem. Although this comprehensive approach results in a block sequencing problem that is more difficult to solve than the ultimate pit limits-based approach, the results are more robust because no block

extraction decisions are made independently. By explicitly incorporating time into the problem formulation, discounting accounts for the time value of money and results in an optimal mine schedule that maximizes the net present value of the ore in the pit.

As mentioned in the introduction, open pit mines are modeled using a collection of three-dimensional rectangular blocks. Each block has a total amount of material and a quantity of valuable mineral associated with it. The goal of the block sequencing problem is determining the proper order in which to remove all the minable blocks in the orebody.

Because, at the most basic level, a mine planner must decide whether or not to remove a given block, we use binary variables to represent the decision regarding whether or not each block in the mine should be removed. To account for its location in the mine, each block is assigned an (x,y,z) position in three-space. We represent the actual time period in which a block is removed by a time index (t). The block sequencing problem formulation is therefore comprised of binary decision variables for each (x,y,z) block in the mine indexed by time (t). Upon solving the problem, each binary variable is assigned a value that indicates during which time period (if at all) each block is scheduled for extraction.

The solution to the block sequencing problem results a value for each binary variable in the problem. If a block is scheduled for removal, then its binary variable with the corresponding time index has a value of one (1). A value of zero (0) means that a block is not removed. Over the life of the mine, many blocks are assigned an extraction date (represented by the time period they are to be removed) and the rest have no extraction date, meaning they are left in the ground.

Binary variables by themselves do not preclude non-integer based solution methods from being employed. However, the presence of *side* constraints such as those imposed by minimum and maximum production, processing, and average grade bounds necessitate the use of mixed integer programming methods to determine the most

efficient block extraction sequence for the mine. Side constraints remove the possibility of using elegant network solution methods and limit us to much slower integer programming approaches.

3.2.1 Two-Dimensional Conceptual Models

To explain our model conceptually, we resort to a basic two-dimensional mine representation (see Figure 3.1). Each block b in the figure has a number used to identify it. Additionally, each block b contains a certain amount of valuable mineral (g_b) and a total amount of material (n_b). The mine's operations are constrained by minimum and maximum bounds with respect to material production rate, ore processing rate, and average extraction grade requirements.

1	**2**	**3**	**4**	**5**	**6**	**7**
8	**9**	**10**	**11**	**12**	**13**	**14**
15	**16**	**17**	**18**	**19**	**20**	**21**

Figure 3.1. Two-Dimensional Conceptual Model. We use this model to describe the basic concepts involved in our problem formulation.

The first issue to resolve in our model is whether each block is *ore* or *waste*. The ore versus waste disposition of each block is determined by comparing the amount of mineral (g_b) to the total amount of material (n_b) in each block. If this ratio is greater than the cutoff grade, then the block is considered *ore*; otherwise, it is considered *waste*. An *ore* block is sent to a processing mill where the valuable mineral is extracted and sold if that block is scheduled for extraction in the final mine schedule. A *waste* block may need to be removed to access other blocks, but it would be disposed of.

Our model assumes a single fixed cutoff grade, meaning that a block is classified

a priori as either *ore* or *waste*. Other models may use a set of cutoff grades. Using a set of cutoff grades means that each block is classified as waste or as a certain quality of ore. Employing a set of cutoff grades means that blocks removed from the mine can be shipped to many different processing mills depending on the quality of ore present in the block. Regardless of whether one or a set of cutoff grades is used, if the cutoff value(s) is (are) fixed, then the ore removed always meets the average grade constraint of the processing mill(s) to which it is sent.

However, if we assume a variable cutoff grade, then *on average* the ore sent to a certain mill must meet that mill's processing requirement. Using a variable cutoff grade necessitates the use of explicit average grade constraints (i.e., minimum and maximum bounds on the average grade of ore sent to each processing mill) as well as another index on each decision variable. This additional index (l) represents the actual location (i.e., the waste dump or a certain processing mill) to which a given block is sent in the optimal solution. Because we assume a fixed cutoff grade, we do not need to concern ourselves with the average grade of the ore sent to the mill; therefore, we do not include average grade constraints in our model formulation.

Sequencing constraints represent the geospatial limitations open pit mines encounter. The basic laws of physics require that the pit is created in such a way so that the resulting hole does not fall in on itself. Commonly referred to as *sloping requirements*, sequencing constraints help prevent such side-wall catastrophes. A commonly assumed slope requirement is 45°, meaning that in order to mine block 9 in Figure 3.1, blocks 1, 2, and 3 must be mined first. Blocks on higher levels in the mine that are not profitable in and of themselves can only be removed if their cost is covered by profitable blocks on lower levels. For instance, if block 1 has a negative profit (i.e., a net cost), then blocks 8 and 9 (which both depend on block 1 being removed to be removed themselves) must have enough net profit to cover the cost of removing block 1. This shared dependence structure in the mine allows high-cost blocks on upper levels to be removed from the pit due to high-value blocks located lower in the

orebody that have a relatively greater profit than the costs of the blocks above.

The slope requirement is enforced throughout the ore-body. To access blocks deeper in the mine, more material above these blocks must be removed. To access block 18 in Figure 3.1, blocks 2, 3, 4, 5, 6, 10, 11 and 12 must all be removed.

Aside from the geospatial constraints, each mine must adhere to a set of operational constraints. Unlike geospatial constraints that do not have a per time period limitation, operational constraints are based on minimum and maximum bounds per time period.

Minimum production bounds require that a certain amount of material is removed from the mine each time period. Likewise, minimum processing bounds require that a certain amount of ore is sent to the mill each time period. Recall that ore is different from material in that ore is material from a block that has met the cutoff grade. These minimum bounds place an upper limit on the life of the mine, a fact that is useful for reducing the number of variables in the problem.

In the same vein, the maximum bounds limit the amount of work done at the mine. With the maximum production capacity and the maximum processing capacity, we can determine a lower bound on the life of the mine. As is the case with the minimum bounds, the maximum bounds are useful for reducing the number of variables in the problem.

3.2.2 Three-Dimensional Conceptual Models

To relate to the real-world, our two-dimensional example must be extended to three dimensions. The sequencing constraints require us to examine a two-dimensional plane of blocks on the level above the block in question. In the two-dimensional example, using a 45° sloping requirement means looking at three blocks above the block in question. When this idea is extended to three dimensions, maintaining the sequencing constraints becomes more difficult.

A common method of satisfying the sloping requirements is to examine the five

blocks that form a plus (+) sign above the block in question (see Figure 3.2). This method satisfies the 45° sloping requirements from the *faces* of the block on the level below.

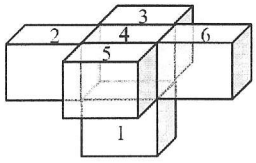

Figure 3.2. 3-D Plus Sign Upper Level Block Example. This graphic depicts the five blocks (blocks 2 through 6) that form a plus (+) sign above the block on the lower level (block 1).

There are other methods of satisfying the sloping requirements, one in particular requiring all nine blocks above the block in question be mined. These nine blocks form a rectangle above the block on the level below (see Figure 3.3). This method satisfies the 45° sloping requirements from the *faces* and the *corners* of the block on the level below. There are other sequencing rules depending on the geography of pit and the surrounding waste rock. For our model, we use the plus sign convention as shown in Figure 3.2.

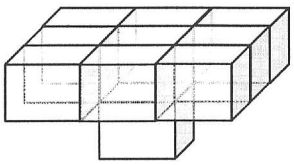

Figure 3.3. 3-D Rectangular Upper Level Block Example. This graphic depicts the nine blocks that form a rectangle above the block on the lower level.

Looking at Figure 3.2, it is apparent that in order to extract block 1 (the lower level block) all five blocks above it must be removed (blocks 2, 3, 4, 5, and 6) to maintain the sloping requirements. Maintaining the sloping requirements in this way satisfies the geospatial sequencing constraints and keeps the hole created by the mining operations from falling in on itself.

The operational constraints are the same for the three-dimensional extension of the problem as they are in the two-dimensional example. Each block is classified as waste or ore based on the assumed cutoff grade and the comparison of mineral content to total material in the block. We must also adhere to production and processing constraints in the same way that we did in the two-dimensional example.

Extending the block sequencing problem to three dimensions significantly increases the number of variables involved in the problem as compared to the two-dimensional conceptual model. Taking the concept even further to a mine composed of thousands or even millions of blocks makes for a truly immense problem. Large deposits requiring higher-fidelity schedules (say, on the order of weeks instead of months or quarters) result in models with millions of binary variables that even the most efficient mixed integer programming algorithms cannot solve in a reasonable amount of time. Problems of this magnitude are more common in today's environment of increasingly scarce natural resources where ore is harder to find and pits must be made bigger and deeper to reach valuable minerals. Solving the block sequencing problem on such a large scale requires more efficient algorithms and stronger formulations to ensure that mine planners optimize their mine's activities.

3.3 Mathematical Formulation

The mathematical formulation for the block sequencing problem is rather straightforward. The objective is to maximize the net present value of the ore in the pit subject to all geospatial and operational constraints. To make the model more tractable, we assume that each block requires only one time period to extract. In other words, once the extraction of a block is started in any given time period, that block is completely removed from the pit by the end of that time period. This also means that a block cannot be extracted over two time periods (i.e., in a fashion such that half the block is removed in one time period and the other half in another time period).

The mathematical formulation also ensures that a block can only be mined once

during the life of the mine. If this limitation is not imposed, the model may attempt to repeatedly extract high valued blocks, leading to erroneous results.

Specifically regarding the constraints imposed on the model, the sequencing and operational constraints ensure that:

- The blocks are removed in a sequence that adheres to geospatial sloping constraints.
- The extraction sequence adheres to a minimum and maximum average grade per time period.[1]
- The extraction sequence adheres to a minimum and maximum production rate per time period.
- The extraction sequence adheres to a minimum and maximum processing rate per time period.

The model formulations (i.e. the *at* formulation in Section 3.3.1 and the *by* formulation in Section 3.3.2) use the following notation:

indices:

b = block

t = time period

sets:

B = set of all blocks (so $b \in B$)

T = set of all time periods, i.e., the time horizon (so $t \in T$)

[1]Note that for our model this is implicitly met due to our fixed cutoff grade assumption. Hence, this constraint is not explicitly required in our formulation, but we include it for completeness.

B_b = set of blocks that must be extracted directly before block b (thus ensuring that the geospatial constraints are adhered to)

T_b = set of time periods in which block b can be extracted

data:

v_{bt} = net present value generated by mining block b in time period t (\$)

g_b = average grade of block b (ounces, tons, percentage)

n_b = total weight of block b including waste (tons)

$$r_b = \begin{cases} n_b & \text{if } g_b \geq \text{ cutoff grade (i.e., if block } b \text{ is classified as } \textit{ore}) \\ 0 & \text{otherwise (i.e, if block } b \text{ is classified as } \textit{waste}) \end{cases}$$

$\underline{G}$ = minimum grade that can be sent to the mill in any time period (ounces, tons, %)

$\overline{G}$ = maximum grade that can be sent to the mill in any time period (ounces, tons, %)

$\underline{C}$ = minimum mill processing requirement in any time period (tons)

$\overline{C}$ = maximum mill processing capacity in any time period (tons)

$\underline{E}$ = minimum mine production requirement in any time period (tons)

$\overline{E}$ = maximum mine production capacity in any time period (tons)

We now clarify some of the notation used above. With respect to the net present value generated by mining block b in time period t (v_{bt}), this number is the net profit realized by removing block b (call it π_b) at time period t discounted by an appropriate discount factor (call it δ). Therefore v_{bt} is calculated as:

$$v_{bt} = \frac{\pi_b}{(1+\delta)^t}$$

A negative net profit represents a block that is not worth removing on its own merit, while a positive net profit represents a block that is worth removing and may be able to pay for the removal of other negative profit blocks above it. With respect to r_b, this number defines the amount of usable ore (i.e., material with enough mineral in it to justify processing it to remove said mineral) contained in block b. If the block has enough mineral content to meet the cutoff grade, then the block (composed of the usable mineral and the waste) is sent to the processing mill where the mineral is removed and sold. However, if the block's mineral content is not sufficient to meet the cutoff grade, then the block is considered waste and its ore weight (r_b) is zero.

3.3.1 *At* Time *t* Formulation

One method of defining the variables for this problem is to identify *at* what time period (the t index) a given block (the b index) is mined. This results in $|T|$ (the number of time periods) variables defined for each block $|B|$ (the total number of blocks in the model), giving us a total of $|T| \cdot |B|$ variables. We present this *at* formulation below ($\mathbf{P}^{\text{at}}$).

Variables: The variables in ($\mathbf{P}^{\text{at}}$) are defined as:

$$y_{bt} = \begin{cases} 1 & \text{if block } b \text{ is mined } at \text{ time period } t \\ 0 & \text{otherwise} \end{cases}$$

Objective Function: The objective of ($\mathbf{P}^{\text{at}}$) is to maximize net present value of the extracted ore:

$$\max \text{ NPV} = \sum_{b \in B} \sum_{t \in T} v_{bt} y_{bt}$$

Constraints: The constraints for ($\mathbf{P}^{\text{at}}$) are:

$$\sum_{t \in T} y_{bt} = 1 \, \forall \, b \ni \arg\max T_b \leq |T| \tag{3.1}$$

$$\sum_{t \in T} y_{bt} \leq 1 \, \forall \, b \ni \arg\max T_b > |T| \tag{3.2}$$

$$\underline{G} \sum_{b \in B} r_b y_{bt} \leq \sum_{b \in B} g_b r_b y_{bt} \leq \overline{G} \sum_{b \in B} r_b y_{bt} \; \forall \, t \tag{3.3}$$

$$\underline{E} \leq \sum_{b \in B} n_b y_{bt} \leq \overline{E} \; \forall \, t \tag{3.4}$$

$$\underline{C} \leq \sum_{b \in B} r_b y_{bt} \leq \overline{C} \; \forall \, t \tag{3.5}$$

$$y_{bt} \leq \sum_{\tau=1}^{t} y_{b'\tau} \; \forall \, b \in B, b' \in B_b, t \tag{3.6}$$

$$y_{bt} \in \{0, 1\} \; \forall \, b, t \tag{3.7}$$

Constraints (3.1) require certain blocks (those that must be mined due to their location in the pit) to be mined during the time horizon. Constraints (3.2) permit certain blocks (those whose location does not require them to be mined during the time horizon) to be mined no more than once during the time horizon. Constraints (3.3) require the average grade of ore extracted to be between a minimum and a maximum in each time period. Constraints (3.4) ensure that the total amount of material extracted is between the minimum and maximum allowable production tonnage. Constraints (3.5) require the total tonnage of ore sent to the mills be between the mills' minimum and maximum processing capacity. Constraints (3.6) are the precedence constraints that enforce extraction sequencing between production blocks based on the 45° plus-sign sloping limitations. For example, according to these sequencing constraints, in order to mine a block with coordinates (x,y,z), the five blocks above this block with coordinates (x,y,z+1), (x-1,y,z+1), (x+1,y,z+1), (x,y-1,z+1), and (x,y+1,z+1) must be mined.[2] Additionally, one of the four blocks on the same level surrounding the given block, i.e., one of the blocks with coordinates (x-1,y,z),

[2]The z-coordinates are labeled such that higher numbers represent blocks higher in the pit.

$(x+1,y,z)$, $(x,y\text{-}1,z)$, or $(x,y+1,z)$, must be mined. We call this constraint the *sixth sequencing constraint* and refer to it as such.[3] Lastly, constraints (3.7) restrict all variables to be binary.

Since our model assumes a fixed cutoff grade, the average grade requirement is met implicitly. As such, the average grade constraints (constraints 3.3) are always met and can be considered redundant and actually removed from the formulation.

3.3.2 *By* Time *t* Formulation

Instead of using binary variables to explicitly define *at* what time period a block is mined, we can define the binary variables to specify whether or not a block is mined *by* a certain time period. This means that the block can be mined in any time period up to and including the one identified by the t index. This idea is first used by Bertsimas and Stock (1998) (also summarized in Bertsimas and Tsitsiklis 1997) with respect to the air traffic flow management problem. Hoffman and Ball (2000) refer to the Bertsimas-Stock idea as a linear transformation in their review of various formulations for the single-airport ground-holding problem. Hoerger, Bachmann, Criss, and Shortridge (1999) are the first to define *by*-based decision variables to formulate their model of long term mine and process scheduling at Newmont's Nevada Operations. Caccetta and Hill (2003) also employ this *by* definition in their formulation of the open pit mine scheduling problem. With this *by* formulation, the decision variables are:

$$w_{bt} = \begin{cases} 1 & \text{if block } b \text{ is mined } by \text{ time period } t \\ 0 & \text{otherwise} \end{cases}$$

The *by* formulation can be translated into the *at* formulation by using the following transformation:

[3] Blocks on the boundary of the orebody are not subject to this constraint.

$$y_{bt} = \begin{cases} w_{bt} - w_{b,t-1} & \text{if } 2 \leq t \leq |T| \\ w_{bt} & \text{if } t = 1 \end{cases}$$

Instances using the *by* formulation tend to solve more quickly than the *at* formulation because the resultant branch-and-bound tree is more balanced. The packing constraints that result with the *at* formulation (constraints 3.1 and 3.2) create a very strong branch (when one of the y_{bt} variables is set equal to 1) and a very weak branch (with the other variables set equal to 0) for a given b and all t. Because of this weak branch, the linear programming (LP) relaxation results in a weak upper bound. When the *by* formulation is used, the two branches are more balanced. The *by* variable set equal to 1 does not provide as strong of a branch as the *at* variable set equal to 1, but the other *by* variables that are set equal to 0 provide a stronger branch than the *at* variables set equal to 0. The resulting better balanced LP relaxation produces a stronger upper bound.

While our *by* formulation is similar to the one described by Bertsimas and Stock (1998), it does not have the same effect. Using the *by* formulation, Bertsimas and Stock create a facet defining constraint set, which results in a problem formulation that is significantly easier to solve. In our case, the resulting constraint set is not facet defining. In fact, we can achieve the same strength of formulation using special ordered sets and the *at* formulation.

We present this alternate formulation using *by*-based decision variables below ($\mathbf{P}^{\text{by}}$).

Variables: The variables in ($\mathbf{P}^{\text{by}}$) are defined as:

$$w_{bt} = \begin{cases} 1 & \text{if block } b \text{ is mined } by \text{ time period } t \\ 0 & \text{otherwise} \end{cases}$$

Objective Function: The objective of ($\mathbf{P}^{\text{by}}$) is to maximize net present value of the extracted ore:

$$\max \text{NPV} = \sum_{b \in B} \sum_{t \in T} v_{bt} \left(w_{bt} - w_{b,t-1} \right)$$

Constraints: The constraints for ($\mathbf{P}^{\text{by}}$) are:

$$w_{b,t-1} \leq w_{bt} \, \forall \, b, t > 1 \tag{3.8}$$

$$\sum_{t \in T} \left(w_{bt} - w_{b,t-1} \right) = 1 \, \forall \, b \ni \arg\max T_b \leq |T| \tag{3.9}$$

$$\sum_{t \in T} \left(w_{bt} - w_{b,t-1} \right) \leq 1 \, \forall \, b \ni \arg\max T_b > |T| \tag{3.10}$$

$$\sum_{b \in B} g_b r_b \left(w_{bt} - w_{b,t-1} \right) \geq \underline{G} \sum_{b \in B} r_b \left(w_{bt} - w_{b,t-1} \right) \ \forall \, t \tag{3.11}$$

$$\sum_{b \in B} g_b r_b \left(w_{bt} - w_{b,t-1} \right) \leq \overline{G} \sum_{b \in B} r_b \left(w_{bt} - w_{b,t-1} \right) \ \forall \, t \tag{3.12}$$

$$\underline{E} \leq \sum_{b \in B} n_b \left(w_{bt} - w_{b,t-1} \right) \leq \overline{E} \ \forall \, t \tag{3.13}$$

$$\underline{C} \leq \sum_{b \in B} r_b \left(w_{bt} - w_{b,t-1} \right) \leq \overline{C} \ \forall \, t \tag{3.14}$$

$$w_{bt} \leq w_{b't} \ \forall \, b \in B, b' \in B_b, t \tag{3.15}$$

$$w_{bt} \in \{0, 1\} \ \forall \, b, t \tag{3.16}$$

Constraints (3.8) require block b to be mined by time period t if it is already mined by time period t-1. Constraints (3.9) require certain blocks (those that must be mined due to their location in the pit) to be mined during the time horizon. Constraints (3.10) permit certain blocks (those whose location does not require them to be mined during the time horizon) to be mined no more than once during the time horizon. Constraints (3.11) and (3.12) require the average grade of ore extracted to be between a minimum (constraints (3.11)) and a maximum (constraints (3.12)) in each

time period.[4] Constraints (3.13) ensure that the total amount of material extracted is between the minimum and maximum allowable production tonnage. Constraints (3.14) require the total tonnage of ore sent to the mills be between the mills' minimum and maximum processing capacity. Constraints (3.15) are precedence constraints that enforce extraction sequencing between production blocks based on the plus-sign sloping limitations. These constraints require that in order to remove block b by time period t, all the blocks above it (i.e., in the set B_b) must be removed by time t also. Lastly, constraints (3.16) restrict all variables to be binary.

As with the *at* formulation, since we assume a fixed cutoff grade, the average grade requirement is met implicitly. Therefore, the average grade constraints (constraints (3.11) and (3.12) above) are always met and can be considered redundant and actually removed from the formulation.

The $\mathbf{P}^{\text{by}}$ constraint set is very similar to the $\mathbf{P}^{\text{at}}$ constraint set, aside from a few notable differences. In the $\mathbf{P}^{\text{by}}$ formulation, constraints (3.8) are added to ensure that once a block is mined by a certain time period, it is also mined every time period thereafter. The sequencing constraints in the $\mathbf{P}^{\text{by}}$ formulation (constraints (3.15)) are also notably different. Specifically, they do not involve any summation. This allows us to compare variable values directly, making for a much stronger formulation.

[4] We split the average grade constraint from the *at* formulation into two separate constraints for the *by* formulation merely for presentation purposes.

Chapter 4

SOLUTION METHODOLOGIES

Since the *by* formulation creates a more balanced node tree than the *at* formulation, we use it to explore all potential methods to expedite solution times. All the methods we examine can also be applied to the *at* formulation, although additional work is required to implement some of these methods due to differences in how the *by* and *at* decision variables are defined.

Exact methods are an approach commonly used to solve mixed integer programming (MIP) problem instances. The branch-and-bound algorithm is the most common exact method used to solve MIP problems. Using branch-and-bound to solve MIP problem formulations guarantees an optimal solution if the algorithm is run to completion. However, this may require a long time, so we often terminate the algorithm and report the gap between the best integer solution found at the time of termination and what may be theoretically obtainable. This gap is referred to as a *mipgap*.

We propose various short-cuts that maintain the exact nature of the branch-and-bound algorithm, but significantly improve its solution time. These short-cuts endeavor to limit the search space by eliminating certain variables or *a priori* setting the values of other variables. Other exact methods involve the generation of valid and useful cuts that make the problem formulation more tractable. Lastly, we use Lagrangian relaxation techniques to solve the problem faster.

Eliminating those variables from the model whose values would necessarily assume a value of 0 or 1 in the optimal solution is one exact method. One way to eliminate variables from the model is to establish an earliest start time for each block $b \in B$ (we call it ES_b). ES_b represents the earliest time period that block b can

be reached if the mining rate occurs at the upper production or processing bound (whichever is tighter) without violating sequencing constraints. The earliest start time allows us to eliminate from the model any variables that would mine block b before its earliest start time (i.e., the values of these variables are set to 0).

Another way to eliminate variables from the model is to establish a latest start time for each block $b \in B$ (we call it LS_b). LS_b represents the latest time period that block b must be mined if the mining rate occurs at the lower production or processing bound (whichever is tighter) without violating sequencing constraints. The latest start time allows us to pre-determine the values of any variable that would mine block b after its latest start time (i.e., the values of these variables are set to 1).

We define our variables only in time periods between their earliest and latest start times. This decreases the number of nodes that the branch-and-bound algorithm must examine, thus speeding up solution times.

It is important to note that determining earliest and latest start times requires that the bounds used in the calculations are fixed (i.e., not elasticized). Some problem instances result in infeasibilities which can be resolved by relaxing some of the constraints. The constraints that are relaxed are *elasticized* by allowing them to be violated at a fixed penalty. If a bound (say, the minimum processing bound) is elasticized, then it cannot be used in the late start calculation.

Exact methods also entail the creation of cuts, or constraints that involve pairs or groups of variables. Cuts are constraints that are added to the formulation that may force the linear relaxation of the problem to behave more like an integer program. The goal is to ensure that none of the cuts eliminates any of the variables in the optimal solution, but do provide a benefit to the algorithm. As such, we aim to create cuts that are valid—so they do not remove any optimal integer solutions—and (theoretically) useful—force decision variables to assume integer values in the linear relaxation of the problem (i.e., strengthen z^*_{LP}). Although many cuts may not be theoretically useful in the strictest sense, they may still provide *practically* useful

information by either limiting the number of blocks that can be accessed by a certain time period or requiring a certain number of blocks be accessed by a certain time period. We exploit the structure of our problem to generate these cuts.

Lagrangian relaxation techniques attempt to transform the monolith problem formulation into a more tractable formulation, solve this new formulation, and then use that solution's decision variable values in the monolith. Specifically, Lagrangian relaxation moves certain *complicating* constraints to the objective function where they are weighted with fixed multipliers to discourage violations. We then solve the Lagrangian relaxation subproblem and use the optimal decision variable values in the monolith (assuming they are feasible), attempting to bound the objective function value of the monolith.

4.1 Numerical Example

To help explain the earliest start, latest start, cut generation, and Lagrangian relaxation methodologies, we use the two-dimensional conceptual model developed in Section 3.2.1. We present this basic two-dimensional mine again below (see Figure 4.1). With respect to this example, we make the following assumptions:

- The numbers in the blocks are merely used to identify the blocks
- Each block contains 10 tons of material (i.e., $n_b = 10$ tons)
- Each block contains 10 grams of valuable mineral (i.e., $g_b = 10$ grams)
- The minimum and maximum processing bounds are 20 tons and 40 tons, respectively, and are constant across all time periods (i.e., $\underline{C} = 20$ and $\overline{C} = 40 \; \forall \, t$)
- The minimum and maximum production bounds are 20 tons and 40 tons, respectively, and are constant across all time periods (i.e., $\underline{E} = 20$ and $\overline{E} = 40 \; \forall \, t$)

- The fixed cutoff grade is 1 gram of mineral per 1 ton of material
$$\left(\text{i.e., } r_b = \begin{cases} n_b & \text{if } g_b \geq \text{ cutoff grade} \\ 0 & \text{otherwise} \end{cases} \right)$$
- The slope requirements are 45°

1	**2**	**3**	**4**	**5**	**6**	**7**
8	**9**	**10**	**11**	**12**	**13**	**14**
15	**16**	**17**	**18**	**19**	**20**	**21**

Figure 4.1. Two-Dimensional Numerical Example. This model provides a numerical example for use in the various methodologies we propose to expedite solution times.

The first issue to resolve in this model is whether each block is *ore* or *waste*. This is determined by applying the cutoff grade to each block. Since each block in the model contains 10 grams of mineral, the average grade of each block is 1 gram of mineral per ton of material. Based on our fixed cutoff grade of 1 gram of mineral per 1 ton of material, each block in this example is considered *ore*. This means that each extracted block is sent to a mill.

The 45° slope requirements mean that the sequencing constraints for this example behave the same way as those described in the two-dimensional conceptual model. See Section 3.2.1 for more information on the sequencing constraints for this example.

Aside from the geospatial constraints, the mine must also adhere to a set of operational constraints. These operational constraints set minimum and maximum bounds on production and processing capacity as well as average grade requirements during each time period. Because this example assumes a fixed cutoff grade, the average grade requirements are rendered moot and do not need to be considered. For simplicity, we assume that both production and processing have the same minimum

and maximum per time period bounds (20 tons and 40 tons, respectively); however, in practice this is rarely the case.

The minimum operational bounds require that a certain amount of work is done at the mine during each time period. The minimum production bound requires that 20 tons of material are removed from the mine each time period. Likewise, the minimum processing bound requires that 20 tons of processable material (i.e., blocks that meet the cutoff grade) are removed from the mine each time period. These minimum bounds place an upper bound on the life of the mine and allow us to establish a maximum time horizon for the mine's operations. Since 20 tons of material and 20 tons of processable material must be removed from the mine each time period, and due to the fact that the mine contains 210 tons of material (21 blocks each with 10 tons of material) and 210 tons of processable material (all 21 blocks meet the cutoff grade, therefore each block is an *ore* block), the life of the mine cannot be longer than 11 time periods (210 divided by 20 is 10.5, so activity at the mine ends during the eleventh time period). We use this fact to reduce the number of variables in the problem via the latest start time idea and to generate cuts.

In the same vein, the maximum operational bounds limit the amount of work done per time period at the mine. With a maximum production capacity of 40 tons of material and a maximum processing capacity of 40 tons of processable material, we can determine a lower bound on the life of the mine (i.e., a minimum time horizon for operations at the mine). Again, since the mine contains 210 tons of material and 210 tons of processable material, if operations are conducted at maximum production and processing levels, the earliest time that activity at the mine can finish is by the sixth time period (210 divided by 40 is 5.25, so activity ceases after the first quarter of the sixth time period). As was the case with the minimum operational bounds, we use the maximum operational bounds to reduce the number of variables in the problem via an earliest start time. We also use these minimum operational bounds to generate cuts.

4.2 Earliest Starts and Latest Starts

We use the concept of earliest and latest starts to eliminate variable instances from consideration and fix variable values, respectively. To determine these starts, we exploit precedence between the blocks along with the minimum and maximum resource bounds; the former bounds produce late starts while the latter bounds produce early starts.

4.2.1 Earliest Starts Based on Maximum Production and Processing Bounds

We can determine an earliest start time for each block $b \in B$, i.e., ES_b, to reduce the number of elements in the set T_b from including the entire time horizon. This earliest start algorithm eliminates those variables from the model whose values would necessarily assume a value of 0 in the optimal solution. By using the sequencing constraints and the upper bounds on production and processing capacity, we determine the earliest possible time that block b can be reached if we were to mine as quickly as possible. We can then eliminate any variables that represent mining block b before its earliest possible start time. Assuming that the upper bounds of neither the production capacity nor the processing capacity are elasticized in our formulation, we determine an earliest start time based on the production capacity and also an earliest start time based on the processing capacity for each block in the model. The overall earliest start time for each block is the later of these two earliest start times. Thus, the tightest earliest start time is established for each block in the model.

Kuchta, Newman, and Topal (2003) use an early start idea with their work at LKAB's Kiruna mine; however, they investigate an underground mine with significantly different sequencing and operational constraints. Their model does not explicitly define maximum bounds on processing or production rates, but instead employs the early start time idea using horizontal and vertical sequencing rules with respect to operating adjacent machine placements (sites on which load haul dump units operate

to remove iron ore from the mine). Boland, Fricke, and Froyland (2006) present a general format for the earliest start method by combining block precedence constraints with production constraints and then aggregating them over time for a particular attribute (such as total ore in each block or amount of usable ore in each block). Our method is similar to theirs; however, we use all applicable attributes (i.e., production and processing capacity limits) to independently determine an earliest start based on each attribute. We then define each block's overall earliest start as the most constraining of the block's independently derived earliest starts based on each attribute.

The earliest start time reflects how long it takes to reach block b based on its location in the pit, the maximum production capacity, and the maximum processing capacity as defined in the problem formulation. Our algorithm, which computes an earliest start time for every block in the pit, first calculates the *support weight* of each block in the pit. This support weight represents the tons of material or the tons of ore from all the blocks that must be mined (based on the sequencing constraints) in order to mine the block in question. The support weight also explicitly includes the tons of material or tons of ore for the given block. We actually calculate two types of support weights, one with respect to tons of material and another with respect to tons of processable material. This support weight with respect to tons of material is then divided by the maximum production capacity and the support weight with respect to tons of processable material is divided by maximum processing capacity to arrive two earliest start times for each block in the pit. Since each block has two earliest start times, the maximum of these two numbers (which represents the later of the two earliest start times) is the earliest possible time period that the block in question can be started.

Earliest Starts Algorithm

Assumptions We include all the assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- B = Set of blocks which exists in the data set. Each block $b \in B$ has the following characteristics:
  - An (x, y, z) location in three-space
  - A total material content (in tons), n_b
  - A mineral content (in grams), g_b
    - If the cutoff grade is met, then the block is considered *ore* and for that block the ore weight (r_b) is:

      $$r_b = n_b$$

    - If the cutoff grade is not met, then the block is considered *waste* and for that block the ore weight (r_b) is:

      $$r_b = 0$$

  - A precedence set – the set of blocks that must be removed from the pit due to pit sloping requirements before block b can be accessed
- S_b = Block b and its precedence set (i.e., the set of blocks that block b *supports*)
- $TotalSupportedOre_b$ = Total amount of ore (in tons) in the set S_b (i.e., block b and all blocks above block b based on the precedence constraints)
- $TotalSupportedMaterial_b$ = Total amount of material (in tons) in the set S_b (i.e., block b and all blocks above block b based on the precedence constraints)

- $EarlyStartOre_b$ = Earliest start time of block b based on the maximum processing constraint
- $EarlyStartMaterial_b$ = Earliest start time of block b based on the maximum production constraint
- ES_b = Earliest start time of block b based on the more constraining bound (processing or production)

Inputs

- A set of blocks B
- Maximum processing capacity per time period (in tons of ore) and maximum production capacity per time period (in tons of material). Note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- ES_b - The earliest possible start time for block $b \in B$ based on geospatial sequencing constraints and the more constraining of the maximum processing and maximum production constraints

Algorithm

begin

for each block $b \in B$ **do**

begin

$TotalSupportedOre_b$ = sum of r_b for each block in S_b (block b and its precedence set)

$TotalSupportedMaterial_b$ = sum of n_b for each block in S_b (block b and its precedence set)

$$EarlyStartOre_b = \left\lfloor \frac{TotalSupportedOre_b}{max\,processing\,capacity} \right\rfloor + 1$$

$$EarlyStartMaterial_b = \left\lfloor \frac{TotalSupportedMaterial_b}{max\,production\,capacity} \right\rfloor + 1$$

$$ES_b = \max\left(EarlyStartOre_b, EarlyStartMaterial_b\right)$$

end

output ES_b for each block $b \in B$

end

Earliest Starts Numerical Example The idea behind the early starts variable elimination routine is best explained by examining Figure 4.2 below:

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.2. Earliest Starts Numerical Example. This example depicts the results of using the earliest starts routine on block 18.

As stated in Section 4.1, the maximum production capacity is 40 tons per period and each block contains 10 tons of material. This means that in order to reach block 18 in the figure above, 80 tons of material have to first be removed based on the assumed 45° sloping requirements. Therefore block 18 cannot be removed until period 3 at the earliest because removing 80 tons of material requires two complete time periods. Hence, block 18 cannot be reached until time period 3. As such, block 18 has an earliest start time of 3 based on the maximum production constraint. Because the maximum processing capacity is also 40 tons per period and each block meets the cutoff grade (so that each block contains 10 tons of processable material), the earliest

start time for block 18 based on maximum processing capacity is also 3. There is no need to investigate the feasibility of block 18 being part of the optimal solution until time period 3. Therefore it is not necessary to define decision variables corresponding to whether or not to mine block 18 during time periods 1 and 2, so, we eliminate these two variables from the problem formulation.

4.2.2 Latest Starts Based on Minimum Production and Processing Bounds

Similar to the concept of earliest start times, we can compute a latest start time for each block $b \in B$, i.e., LS_b. Generating a latest start time for each block forces the algorithm to set each block's value to *mined* upon reaching its latest start time by fixing its value to 1 in the optimal solution. Using the sequencing constraints in conjunction with the lower bounds on production and processing capacity, we determine the latest possible time that block b can be reached if we were to mine as slowly as possible. We can set the value of any variable that would indicate mining block b after its latest start time to 1 (i.e., *mined*). Assuming that the lower bounds of neither the production requirement nor the processing requirement are elasticized in our formulation, we determine a latest start time based on production requirements and processing requirements for each block in the model. The overall latest start time for each block is the earlier of these two latest start times. Therefore, the tightest latest start time is established for each block in the model.

Kuchta, Newman, and Topal (2003) employ the latest start idea in their work with LKAB's Kiruna mine. However, their model does not explicitly define minimum bounds on processing or production rates. Instead, they use horizontal and vertical sequencing rules regarding adjacent machine placements along with information regarding which machine placements are active. Boland, Fricke, and Froyland (2006) do not present any methodology for generating latest starts. Their work only covers earliest starts.

The latest start time reflects the most time that can pass before block b must

be mined based on its location in the pit, the minimum production requirement, and the minimum processing requirement as defined in the problem formulation. Our algorithm computes a latest start time for every block in the pit by determining each block's *holding weight*. This holding weight represents the tons of material or the tons of ore that cannot be mined (based on the sequencing constraints) until the block in question is mined. The holding weight also explicitly includes the tons of material or tons of ore for the block in question. Just as with the earliest start times concept, we actually calculate two types of holding weights, one with respect to tons of material and another with respect to tons of processable material. The holding weight with respect to tons of material is then divided by the minimum production requirement while the holding weight with respect to tons of processable material is divided by the minimum processing requirement to arrive at two latest start times for each block in the pit. The overall latest start time for each block is the earlier of the two latest start times calculated for each block.

Latest Starts Algorithm

Assumptions Just as with the earliest starts algorithm, we include all the assumptions that we describe with respect to our model formulation (see Section **??**).

Definitions

- B represents the set of blocks which exists in the data set. Each block $b \in B$ has the following characteristics:
  - An (x, y, z) location in three-space
  - A total material content (in tons), n_b
  - A mineral content (in grams), g_b
    - If the cutoff grade is met, then the block is considered *ore* and for that block the ore weight (r_b) is:

$r_b = n_b$

* If the cutoff grade is not met, then the block is considered *waste* and for that block the ore weight (r_b) is:

$r_b = 0$

– A holding set – the set of blocks that is being *held up* by block b (i.e., all the blocks in B that cannot be removed from the pit due to pit sloping requirements until block b is removed from the pit)

- H_b = Block b and its holding set (i.e., the set of blocks being *held up* by block b from being mined)
- $TotalHeldUpOre_b$ = Total amount of ore (in tons) in the set H_b (i.e., block b and all blocks below block b based on the precedence constraints)
- $TotalHeldUpMaterial_b$ = Total amount of material (in tons) in the set H_b (i.e., block b and all blocks below block b based on the precedence constraints)
- $TotalOreInPit$ = Total amount of ore (in tons) in the entire pit (sum of all ore blocks in B, i.e., sum of r_b for all $b \in B$)
- $TotalMaterialInPit$ = Total amount of material (in tons) in the entire pit (sum of all material in B, i.e., sum of n_b for all $b \in B$)
- $LateStartOre_b$ = Latest start time of block b based on the minimum processing constraint
- $LateStartMaterial_b$ = Latest start time of block b based on the minimum production constraint
- LS_b = Latest start time of block b based on the most constraining bound (processing or production)

Inputs

- A set of blocks **B**
- Minimum processing requirements per time period (in tons of ore) and minimum production requirements per time period (in tons of material). Note that these requirement constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- LS_b - The latest possible start time for block $b \in B$ based on geospatial sequencing constraints and the more constraining of the minimum processing and minimum production constraints

Algorithm

begin

for each block $b \in B$ **do**

begin

$TotalHeldUpOre_b$ = sum of r_b for each block in H_b (block b and its holding set)

$TotalHeldUpMaterial_b$ = sum of n_b for each block in H_b (block b and its holding set)

$LateStartOre_b = \left\lfloor \frac{TotalOreInPit - TotalHeldUpOre_b}{min\,processing\,requirement} \right\rfloor + 1$

$LateStartMaterial_b = \left\lfloor \frac{TotalMaterialInPit - TotalHeldUpMaterial_b}{min\,production\,requirement} \right\rfloor + 1$

$LS_b = \min\left(LateStartOre_b, LateStartMaterial_b\right)$

end

output LS_b for each block $b \in B$

end

Latest Starts Numerical Example We now present a numerical example of the latest starts variable elimination routine by examining Figure 4.3 below:

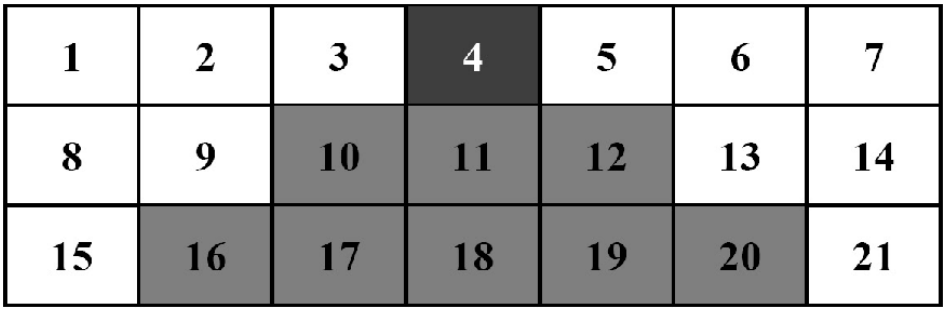

Figure 4.3. Latest Starts Numerical Example. This example depicts the results of using the latest starts routine on block 4.

According to the figure above, block 4 is preventing blocks 10, 11, 12, 16, 17, 18, 19, and 20 from being mined. This only leaves blocks 1, 2, 3, 5, 6, 7, 8, 9, 13, 14, 15, and 21 to be mined before block 4 must be mined. As stated in Section 4.1, each block contains 10 tons of material. The entire pit contains 210 tons of material and block 4 is holding up production of 90 tons of material, so the leftover material that can be mined is 120 tons of material ($210 - 90 = 120$). Based on the assumed minimum production capacity of 20 tons per period, the latest that block 4 can be started is time period 7 (120 divided by 20 is 6, so block 4 must start being mined at the beginning of time period 7). Since the minimum processing capacity is also 20 tons per period and each block meets the cutoff grade (so that each block contains 10 tons of processable material), the latest start time for block 4 based on minimum processing capacity is also 7. As such, block 4 has a latest start time of 7. The value of the decision variables for block 4 during any time period including and after time period 7 must be 1 (recall, 1 means that the block is considered *mined*). As a result of the latest start routine, we can *a priori* set the values of block 4's decision variables after time period 6 in our model formulation to 1.

4.3 Cut Generation Techniques

Generating cuts involves creating valid and useful inequalities that define pairs or sets of blocks that cannot be mined together. Cuts are constraints that are added to the formulation that may force the linear relaxation of the problem to behave more like an integer program. These constructed cuts must not eliminate any optimal integer solutions (i.e., they must be *valid*) and should strengthen the formulation by forcing decision variables to assume integer values in the LP relaxation of the problem or eliminating the optimal LP relaxation solution (i.e., they should be *useful*). We exploit the structure of our problem to create cuts that are valid and useful inequalities in the form of packing constraints ($\leq$) and covering constraints ($\geq$).

4.3.1 Cuts in General

All generated cuts must be valid and should be useful. By valid, we mean that the cut cannot remove any feasible integer solutions. If a cut is not valid, then its inclusion in the problem formulation may result in a sub-optimal solution. Regarding usefulness, there is a difference between *theoretical* usefulness and *practical* usefulness. Theoretically, a cut is considered useful if, among other things, it renders infeasible the optimal solution to the current LP relaxation (Rardin 1998, p. 644). From a practical standpoint, however, a cut that is not theoretically useful may still make the model formulation more tractable. For instance, consider a cut that states that at most one of two binary variables can assume a value of one (i.e., a cut in the form of $\mathbf{a} + \mathbf{b} \leq 1$, where $\mathbf{a}$ and $\mathbf{b}$ both represent binary variables). Such a cut may not be theoretically useful, but from a practical standpoint, it may be very useful (depending, of course, on the other constraints in the model which may render the cut redundant). If we know that one variable (say $\mathbf{a}$) has a value of 1, then without any further computation we also know that the other variable must have a value of 0 (i.e., $\mathbf{b} = 0$). The practical usefulness of these cuts may help the branch-and-bound

process to solve our mixed integer programming problem more quickly.

We use a *reasonable block selection rule* to determine which blocks to investigate for the generation of cuts. Although any pair or set of blocks can be combined to form a cut, many cuts created in such an arbitrary manner are not useful (theoretically or practically). As such, it is important to limit the number of blocks that are used to create cuts. We wish to pick the *best* blocks to investigate for cut generation, thus increasing the likelihood that the blocks create a valid and useful cut.

Our *reasonable block selection rule* eliminates from contention those blocks which have little chance of creating a valid and useful cut. The rule examines individual blocks while the cut generation algorithms involve multiple blocks. As such, our cut generation procedures ensure that the cuts are valid and are at least practically useful, while our *reasonable block selection rule* dictates which blocks the cut generation procedures investigate. We employ the reasonable block selection rule as a means of picking the best blocks for inclusion in cuts, realizing that this rule may actually eliminate some blocks that could form a valid and useful cut. However, the only way to create every valid and useful cut is to investigate every possible combination of blocks; a task that is computationally too expensive.

To derive such a *reasonable block selection rule*, we borrow the support weight and holding weight ideas explained in the earliest and latest start routines. These weights allow us to intelligently select blocks for use in creating cuts that are valid and have a good chance of being useful. We do this by investigating only those blocks whose supporting weight (or holding weight) is within a certain percent of the block's next earliest start weight (or latest start weight). For earliest starts cuts, this rule is:

$$PercentClose_{grade} = \frac{wt_{ore}}{ES * MaxProc}$$

$$PercentClose_{material} = \frac{wt_{material}}{ES * MaxProd}$$

where wt_{ore} and $wt_{material}$ represent the processable material weight and total material

weight of the block and all of its predecessors, respectively; *ES* is the earliest start for the block; and *MaxProc* and *MaxProd* are the maximum processing and production capacities per time period, respectively. For latest starts cuts, the rule is slightly different:

$$PercentClose_{grade} = \frac{wt_{ore}}{TotalOreInPit - ((LS-1) * MinProc)}$$

$$PercentClose_{material} = \frac{wt_{material}}{TotalMaterialInPit - ((LS-1) * MinProc)}$$

where wt_{ore} and $wt_{material}$ represent the processable material weight and total material weight of the block and all of its holders, respectively (i.e., blocks that are being *held up*, or prevented from being mined, due to the block in question); *LS* is the latest start for the block; *TotalOreInPit* and *TotalMaterialInPit* represent the total amount of ore and total amount of material in the entire data set; and *MinProc* and *MinProd* are the minimum processing and production requirements per time period, respectively. We explain this rule with some examples.

By *next earliest start weight* we mean the amount of production or processing capacity (whichever is smaller) required to push the block's earliest start time to the very next time period. For instance, in the numerical example we have been employing, the maximum production capacity is 40 tons per period. Let us identify a block, **a**, with a support weight of 10 tons. This block has an earliest start of 1 $\left(\lfloor \frac{10}{40} \rfloor + 1 = 1\right)$ and is only 25% close to its next earliest start weight $\left(\frac{10}{1*40} \times 100\% = 25\%\right)$. As such, block **a** would not be a good candidate for use in generating a cut based on our reasonable block selection rule. Let us identify another block, **b**, with a support weight of 35 tons. This block also has an earliest start of 1 $\left(\lfloor \frac{35}{40} \rfloor + 1 = 1\right)$, but it is 87.5% close to its next earliest start weight $\left(\frac{35}{1*40} \times 100\% = 87.5\%\right)$. Block **b** would be a much better candidate for inclusion in a cut than block **a**. Note, however, that a cut formed by combining blocks **a** and **b** actually would form a valid cut of the form $w_{\mathbf{a},1} + w_{\mathbf{b},1} \leq 1$ because, assuming they share no blocks between their respective

predecessor sets, together these two blocks cannot both be mined in time period 1 (because their combined weight is 45 tons and only 40 tons can be mined in time period 1).

Similarly, by *next latest start weight* we mean the amount of production or processing requirement (whichever is larger) left to push the block's latest start time to the previous time period. Again, we consider our numerical example where minimum production capacity is 20 tons per period. We assume our pit contains 210 tons of material (consistent with the 21 blocks in our example, each weighing 10 tons). Consider a block, call it **a**, with a holding weight of 12 tons. This block has a latest start of 10 $\left(\left\lfloor\frac{210-12}{20}\right\rfloor + 1 = 10\right)$ and is 40% close to its next latest start weight $\left(\frac{12}{210-((10-1)*20)} \times 100\% = 40\%\right)$. Block **a** probably would not be a good candidate for use in generating a cut. Let us look at another block, **b**, with a holding weight of 28. This block also has a latest start of 10 $\left(\left\lfloor\frac{210-28}{20}\right\rfloor + 1 = 10\right)$ but it is 93.3% close to its next latest start weight $\left(\frac{28}{210-((10-1)*20)} \times 100\% = 93.3\%\right)$. Picking among these two blocks, block **b** would be the better candidate to include in a set of potential blocks for latest start cut generation.

The *percent close* numbers here are to illustrate the procedure only. Ultimately, the user defines the percentage above which a block passes the *reasonable block selection rule*. A higher percentage reduces the number of blocks included in the cut generation procedure. Our *reasonable block selection rule* considers individual blocks, but the generation of cuts involves two or more blocks. Investigating all two-way, three-way, etc. combinations of blocks would quickly become computationally too expensive. As such, we use our *reasonable block selection rule* as a proxy to select the best individual blocks to include in the generation of valid and useful multi-block cuts.

Despite limiting the number of blocks we investigate with our cut generation algorithm, we still examine many combinations of blocks. Examining all of these block combinations takes a long time, especially as the number of blocks in the cut

increases (i.e., there are fewer two-way combinations of a given set of blocks than there are three-way combinations of the same set of blocks). The time saved by employing these cuts in our model formulation might be lost due to the time spent actually creating them. We want to ensure that the reduction in solution time in our numerical results is not offset by the time required to create the cuts. Determining the amount of time to spend generating cuts involves a degree of judgment and must be balanced with the time it takes to solve the monolith. We explore this more in our numerical results (Section 5.3.2).

Boland, Fricke, and Froyland (2006) present a method of generating cuts by defining valid knapsack inequalities to serve as cover cuts. However, they only discuss cuts of the form:

$$\sum_{b \in B} w_{bt} \leq |\hat{B}| - 1$$

where $\hat{B}$ is the set of blocks involved in the cut (i.e., $\hat{B} = \{\mathbf{a}, \mathbf{b}\}$ for the two-block examples above using our reasonable block selection rule). We describe cuts of the form:

$$\sum_{b \in B} w_{bt} \geq |\hat{B}| - 1$$

also, which Boland, Fricke, and Froyland do not address. Additionally, they do not employ a *reasonable block selection rule* to select blocks for cut generation, instead attempting to generate cuts using all available blocks.

The right-hand-side of cuts involving more than two blocks can have values up to one fewer than the number of blocks involved in the cut (i.e., $\left\{1, 2, \ldots, |\hat{B}| - 1\right\}$). The cuts generated by Boland, Fricke, and Froyland, however, only permit a right-hand-side that is exactly *equal to* the number of blocks involved in the cut minus one (i.e., they only use a right-hand-side that equals $\left(|\hat{B}| - 1\right)$). Our cuts are not limited in this way. These added cuts further speed up solution times.

To generate cuts involving multiple blocks, we investigate super-blocks, which are formed by combining the blocks in question and their respective precedence sets (or

holding sets, as the case may be). When creating these super-blocks, it is important to use the union operator so that no blocks are double counted in the combined set. If we consider blocks **a** and **b**, then the union of their precedence sets (i.e., $S_{\mathbf{a},\mathbf{b}}$) contains the blocks in **a**'s precedence set ($S_{\mathbf{a}}$) and the blocks in **b**'s precedence set ($S_{\mathbf{b}}$), without any shared blocks between the two sets counted more than once (i.e., $S_{\mathbf{a},\mathbf{b}} = S_{\mathbf{a}} \cup S_{\mathbf{b}}$). We then use this super-block in our earliest starts algorithm (or latest starts algorithm, as the case may be) to determine the earliest possible time that both blocks **a** and **b** can be accessed together as a unit. The earliest starts algorithm uses the super-block $S_{\mathbf{a},\mathbf{b}}$ instead of the precedence sets $S_{\mathbf{a}}$ and $S_{\mathbf{b}}$ in all the calculations. The same idea holds for super-blocks formed by combining three or more blocks and for super-blocks used to determine latest starts.

The cuts generation algorithms allows the user to create cuts based on either production bounds, processing bounds, or both. As mentioned above, the user controls the *percent close* employed by the *reasonable block selection rule*. The user also controls whether the algorithm generates *theoretically* useful cuts, *practically* useful cuts, or both.

4.3.2 Two-Way Earliest Starts Cuts

A potentially valid and useful cut for our model involves allowing at most one of two blocks to be mined by a particular time period due to the maximum production and/or maximum processing constraint. Because we are allowing at most one of two blocks to be mined, this cut takes the following form:

$$w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$$

where **a** and **b** are arbitrarily chosen blocks that adhere to the *reasonable block selection rule* (see Section 4.3.1). Generate the cut by comparing the earliest start time for each individual block ($ES_{\mathbf{a}}$ and $ES_{\mathbf{b}}$, respectively) with the earliest start time of

the super-block formed by the union of blocks **a** and **b** ($ES_{\mathbf{a},\mathbf{b}}$, referred to as τ). If τ is greater than both of the single block earliest start times (i.e., $\tau > \max(ES_{\mathbf{a}}, ES_{\mathbf{b}})$), then the super-block formed by the union of blocks **a** and **b** can only be accessed by a time period later than the earliest start times for the individual blocks. Because of this, access is limited to only one of these two blocks by time period $\tau - 1$, and an appropriate cut of the form $w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$ can be generated.[1]

Determining if the Two-Way Earliest Starts Cuts are Valid and Useful Only cuts that are valid and useful should be included in the model formulation. To determine if our cuts of the form $w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$ meet these criteria, we must pay particular attention to the time index $\tau - 1$.

Based on the earliest start of the super-block formed by the union of blocks **a** and **b** ($ES_{\mathbf{a},\mathbf{b}}$, which we call τ), we know that τ is the earliest possible time period that both blocks **a** and **b** can be mined together. This means that during any time period before τ, only one of these two blocks can be mined. So, it is valid to limit access to at most one of the two blocks **a** and **b** by time period $\tau - 1$.

It may be *practically* useful to limit at most one of these two blocks **a** and **b** to be accessed by time period $\tau - 1$, because if the value of one of the blocks is known to be *mined* (i.e., say $w_{\mathbf{a},\tau-1} = 1$) then the value of the other block is also known due to the cut ($w_{\mathbf{b},\tau-1}$ must equal 0 or the constraint represented by the cut is violated). To determine the *theoretical* usefulness of the cut, however, we must empirically test each cut. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For two-way earliest starts cuts of the form $w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\tau-1}$ and $w_{\mathbf{b},\tau-1}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\tau-1}$ and $\tilde{w}_{\mathbf{b},\tau-1}$) is

[1]Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block and the t index identifies a time period by which the block is extracted.

greater than 1:

$$\tilde{w}_{\mathbf{a},\tau-1} + \tilde{w}_{\mathbf{b},\tau-1} > 1$$

Two-Way Earliest Starts Cuts Algorithm

Assumptions We again use all assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- $\mathbf{a}$ = A block (from the set of blocks B) which adheres to the *reasonable block selection rule*
- $\mathbf{b}$ = Another block (from the set of blocks B and not the same as $\mathbf{a}$) which adheres to the *reasonable block selection rule*
- $S_{\mathbf{a},\mathbf{b}}$ = Set of blocks that must be mined (based on the sequencing constraints) in order to mine blocks $\mathbf{a}$ and $\mathbf{b}$ (including explicitly mining blocks $\mathbf{a}$ and $\mathbf{b}$). This set contains blocks $\mathbf{a}$ and $\mathbf{b}$ and the union of all the blocks in each of their respective precedence sets (i.e., $S_{\mathbf{a},\mathbf{b}} = S_{\mathbf{a}} \cup S_{\mathbf{b}}$, since $S_{\mathbf{a}}$ contains block $\mathbf{a}$ and all the blocks in block $\mathbf{a}$'s precedence set and $S_{\mathbf{b}}$ contains block $\mathbf{b}$ and all the blocks in block $\mathbf{b}$'s precedence set). As a result, no shared blocks between the precedence sets of blocks $\mathbf{a}$ and $\mathbf{b}$ are counted more than once in the super-block represented by $S_{\mathbf{a},\mathbf{b}}$.
- $ES_{\mathbf{a}}$ = Earliest start time for block $\mathbf{a}$ (based on either the maximum processing constraint or the maximum production constraint)
- $ES_{\mathbf{b}}$ = Earliest start time for block $\mathbf{b}$ (based on either the maximum processing constraint or the maximum production constraint)

- $\tau = ES_{\mathbf{a},\mathbf{b}}$ = Earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b}}$ (based on either the maximum processing constraint or the maximum production constraint)

Inputs

- A set of blocks B
- Maximum processing capacity per time period (in tons of ore) and maximum production capacity per time period (in tons of material). Note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- Valid cuts of the form:

$$w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$$

Algorithm

For each two-way combination of blocks $\mathbf{a} \in B$ and $\mathbf{b} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the maximum *production* capacity **do**:

1a. Determine the earliest start time for block **a** (i.e., $ES_{\mathbf{a}}$) based on the maximum production capacity

2a. Determine the earliest start time for block **b** (i.e., $ES_{\mathbf{b}}$) based on the maximum production capacity

3a. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **b** (i.e., $S_{\mathbf{a},\mathbf{b}}$)

4a. Determine the earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b}}$ (i.e., $\tau = ES_{\mathbf{a},\mathbf{b}}$) based on the maximum production capacity

5a. If $\tau > \max(ES_{\mathbf{a}}, ES_{\mathbf{b}})$ then create a cut of the form:

$$w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$$

For each two-way combination of blocks $\mathbf{a} \in B$ and $\mathbf{b} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the maximum *processing* capacity **do**:

1b. Determine the earliest start time for block $\mathbf{a}$ (i.e., $ES_{\mathbf{a}}$) based on the maximum processing capacity

2b. Determine the earliest start time for block $\mathbf{b}$ (i.e., $ES_{\mathbf{b}}$) based on the maximum processing capacity

3b. Create the set of blocks that represents the union of the precedence sets for blocks $\mathbf{a}$ and $\mathbf{b}$ (i.e., $S_{\mathbf{a},\mathbf{b}}$)

4b. Determine the earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b}}$ (i.e., $\tau = ES_{\mathbf{a},\mathbf{b}}$) based on the maximum processing capacity

5b. If $\tau > \max(ES_{\mathbf{a}}, ES_{\mathbf{b}})$ then create a cut of the form:

$$w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1$$

Output all generated cuts

Relative Dominance of Two-Way Earliest Starts Cuts The time index (τ) in the cuts we generate provides information about the relative dominance of different cuts. Take the following two potential cuts:

- $w_{\mathbf{a},2} + w_{\mathbf{b},2} \leq 1$
- $w_{\mathbf{a},4} + w_{\mathbf{b},4} \leq 1$

The first means that at most one of the two blocks **a** and **b** can mined by time period 2, while the second means that at most one of these two blocks can be mined by time period 4. In this case the latter cut dominates the former. The reason for this dominance is analogous to the reason that an earliest start of 4 is a stronger restriction than an earliest start of 2 for any given block. Our two-way earliest starts cuts algorithm accounts for this dominance and only generates the dominant cut for any given pair of blocks (assuming such a cut is valid and useful).

Two-Way Earliest Starts Cuts Numerical Example Looking at our two-dimensional example, we use blocks 10 and 12 to create a two-way earliest starts cut based on the maximum production capacity (see Figure 4.4).

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.4. Two-Way Earliest Starts Cuts Numerical Example. This example depicts the results of creating a two-way earliest starts cut with blocks 10 and 12.

For this example, we assume that:

- Block **a** is represented by block 10 in the figure and block **b** is represented by block 12 in the figure
- Each block contains 10 tons of material (i.e., $n_b = 10$ for each block **a** and **b**)
- The maximum production capacity is 40 tons per time period (for simplicity, we only use production bounds for this example)

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block (blocks 10 and 12 in the figure above for this example) and the t index identifies the time period by which the block is extracted. We now use the algorithm to generate a two-way earliest starts cut based on maximum production capacity:

1. Determine the earliest start time for block **a**:

   $ES_{\mathbf{a}} = 1$

2. Determine the earliest start time for block **b:**

   $ES_{\mathbf{b}} = 1$

3. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **b**:

   $S_{\mathbf{a},\mathbf{b}} = S_{\mathbf{a}} \cup S_{\mathbf{b}} = \{2, 3, 4, 10\} \cup \{4, 5, 6, 12\} = \{2, 3, 4, 5, 6, 10, 12\}$

4. Determine the earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b}}$:

   $\tau = ES_{\mathbf{a},\mathbf{b}} = 2$

5. Since $\tau > \max(ES_{\mathbf{a}}, ES_{\mathbf{b}})$ we can create a cut of the form:

   $w_{\mathbf{a},\tau-1} + w_{\mathbf{b},\tau-1} \leq 1 \qquad \Rightarrow \qquad w_{10,1} + w_{12,1} \leq 1$

This means that by the end of time period 1, at most one of the two blocks represented by block 10 and block 12 in the figure above can be mined.

4.3.3 Two-Way Latest Starts Cuts

Building on the two-way earliest starts cuts idea, we now present the latest starts version of that cut. This potentially valid and useful cut involves forcing at least one of two blocks to be mined by a particular time period due to the minimum production

and/or minimum processing constraint. Since we are forcing at least one of two blocks to be mined, this cut takes the following form:

$$w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$$

where **a** and **b** are arbitrarily chosen blocks that adhere to the *reasonable block selection rule* (see Section 4.3.1). The cut is generated by comparing the latest start time for each individual block ($LS_{\mathbf{a}}$ and $LS_{\mathbf{b}}$, respectively) with the latest start time of the super-block formed by the union of blocks **a** and **b** ($LS_{\mathbf{a},\mathbf{b}}$, referred to as $\hat{\tau}$). If $\hat{\tau}$ is less than both of the single block earliest start times (i.e., $\hat{\tau} < \min(LS_{\mathbf{a}}, LS_{\mathbf{b}})$), then the super-block formed by the union of blocks **a** and **b** must be accessed by a time period earlier than the latest start times for the individual blocks. Because of this, one of these two blocks must be extracted by time period $\hat{\tau}$, and an appropriate cut of the form $w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$ can be generated.[2]

Determining if the Two-Way Latest Starts Cuts are Valid and Useful As with the earliest starts cuts, only valid and useful cuts should be included in the model formulation. To determine if our cuts of the form $w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$ meet these criteria, we again pay particular attention to the time index, which is $\hat{\tau}$ in this case.

Based on the latest start of the super-block formed by the union of blocks **a** and **b** ($LS_{\mathbf{a},\mathbf{b}}$, which we call $\hat{\tau}$), we know that $\hat{\tau}$ is the latest possible time period that both blocks **a** and **b** must be mined because as a unit they are holding up access to the remaining blocks in the pit. By time period $\hat{\tau}$, therefore, at least one of these two blocks must be removed from the pit. Even if there exists a single block among **a** and **b** that does not need to be extracted (on its own) until a time period later than $\hat{\tau}$, we still need to remove at least one of the two blocks **a** or **b** by time period $\hat{\tau}$ to meet the minimum production and/or minimum processing constraints.

[2]Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block and the t index identifies a time period by which the block is extracted.

It may be *practically* useful to require at least one of these two blocks **a** and **b** be mined by time period $\hat{\tau}$, because if the value of one of the blocks is determined to be *not mined* (i.e., say $w_{\mathbf{a},\hat{\tau}} = 0$) then the value of the other block is also known due to the cut ($w_{\mathbf{b},\hat{\tau}}$ must equal 1 or the constraint represented by the cut is violated). Just as with the earliest starts cuts, to determine the *theoretical* usefulness of the cut, we must empirically test each cut. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For two-way latest starts cuts of the form $w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\hat{\tau}}$ and $w_{\mathbf{b},\hat{\tau}}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\hat{\tau}}$ and $\tilde{w}_{\mathbf{b},\hat{\tau}}$) is less than 1:

$$\tilde{w}_{\mathbf{a},\hat{\tau}} + \tilde{w}_{\mathbf{b},\hat{\tau}} < 1$$

Two-Way Latest Starts Cuts Algorithm

Assumptions We again use all assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- **a** = A block (from the set of blocks B) which adheres to the *reasonable block selection rule*
- **b** = Another block (from the set of blocks B and not the same as **a**) which adheres to the *reasonable block selection rule*
- $H_{\mathbf{a},\mathbf{b}}$ = Set of blocks that cannot be mined (based on the sequencing constraints) until blocks **a** and **b** are mined (including explicitly mining blocks **a** and **b**). This set contains blocks **a** and **b** and the union of all the blocks in each of their respective holding sets (i.e., $H_{\mathbf{a},\mathbf{b}} = H_{\mathbf{a}} \cup H_{\mathbf{b}}$, since $H_{\mathbf{a}}$ contains block **a** and all the blocks in block **a**'s holding set and $H_{\mathbf{b}}$ contains block **b** and all the blocks

in block **b**'s holding set). As a result, no shared blocks between the holding sets of blocks **a** and **b** are counted more than once in the super-block represented by $H_{\mathbf{a},\mathbf{b}}$.

- $LS_{\mathbf{a}}$ = Latest start time for block **a** (based on either the minimum processing constraint or the minimum production constraint)
- $LS_{\mathbf{b}}$ = Latest start time for block **b** (based on either the minimum processing constraint or the minimum production constraint)
- $\hat{\tau} = LS_{\mathbf{a},\mathbf{b}}$ = Latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b}}$ (based on either the minimum processing constraint or the minimum production constraint)

Inputs

- A set of blocks B
- Minimum processing requirement per time period (in tons of ore) and minimum production requirement per time period (in tons of material). Note that these requirement constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- Valid cuts of the form:

$$w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$$

Algorithm

For each two-way combination of blocks $\mathbf{a} \in B$ and $\mathbf{b} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the minimum *production* requirement **do**:

1a. Determine the latest start time for block **a** (i.e., $LS_{\mathbf{a}}$) based on the minimum production requirement

2a. Determine the latest start time for block **b** (i.e., $LS_{\mathbf{b}}$) based on the minimum production requirement

3a. Create the set of blocks that represents the union of the holding sets for blocks **a** and **b** (i.e., $H_{\mathbf{a},\mathbf{b}}$)

4a. Determine the latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b}}$ (i.e., $\hat{\tau} = LS_{\mathbf{a},\mathbf{b}}$) based on the minimum production requirement

5a. If $\hat{\tau} < \min(LS_{\mathbf{a}}, LS_{\mathbf{b}})$ then create a cut of the form:

$$w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$$

For each two-way combination of blocks $\mathbf{a} \in B$ and $\mathbf{b} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the minimum *processing* requirement **do**:

1b. Determine the latest start time for block **a** (i.e., $LS_{\mathbf{a}}$) based on the minimum processing requirement

2b. Determine the latest start time for block **b** (i.e., $LS_{\mathbf{b}}$) based on the minimum processing requirement

3b. Create the set of blocks that represents the union of the holding sets for blocks **a** and **b** (i.e., $H_{\mathbf{a},\mathbf{b}}$)

4b. Determine the latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b}}$ (i.e., $\hat{\tau} = LS_{\mathbf{a},\mathbf{b}}$) based on the minimum processing requirement

5b. If $\hat{\tau} < \min(LS_{\mathbf{a}}, LS_{\mathbf{b}})$ then create a cut of the form:

$$w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1$$

Output all generated cuts

Relative Dominance of Two-Way Latest Starts Cuts As with the two-way earliest start cuts, the time index ($\hat{\tau}$) in the cuts we generate provides information about the relative dominance of different cuts. Take the following two potential cuts:

- $w_{\mathbf{a},2} + w_{\mathbf{b},2} \geq 1$
- $w_{\mathbf{a},4} + w_{\mathbf{b},4} \geq 1$

The first means that at least one of the two blocks **a** and **b** must mined by time period 2, while the second means that at least one of these two blocks must be mined by time period 4. In this case, the former cut dominates the latter. The reason for this dominance is analogous to the reason that a latest start of 2 is a stronger restriction than a latest start of 4 for any given block. Our two-way latest starts cuts algorithm accounts for this dominance and only generates the dominant cut for any given pair of blocks (assuming such a cut is valid and useful).

Two-Way Latest Starts Cuts Numerical Example Looking at our two-dimensional example, we use blocks 10 and 12 to create a two-way latest starts cut based on the minimum production requirement (see Figure 4.5).

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.5. Two-Way Latest Starts Cuts Numerical Example. This example depicts the results of creating a two-way latest start cut with blocks 10 and 12.

For this example, we assume that:

- Block **a** is represented by block 10 in the figure and block **b** is represented by block 12 in the figure
- Each block contains 10 tons of material (i.e., $n_b = 10$ for each block **a** and **b**)
- The minimum production requirement is 20 tons per time period (for simplicity, we only use production bounds for this example)

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block (blocks 10 and 12 in the figure above for this example) and the t index identifies the time period by which the block is extracted. We now use the algorithm to generate a two-way latest starts cut based on the minimum production requirement:

1. Determine the latest start time for block **a**:

   $LS_{\mathbf{a}} = 9$

2. Determine the latest start time for block **b:**

   $LS_{\mathbf{b}} = 9$

3. Create the set of blocks that represents the union of the holding sets for blocks **a** and **b**:

   $H_{\mathbf{a},\mathbf{b}} = H_{\mathbf{a}} \cup H_{\mathbf{b}} = \{10, 16, 17, 18\} \cup \{12, 18, 19, 20\} = \{10, 12, 16, 17, 18, 19, 20\}$

4. Determine the latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b}}$:

   $\hat{\tau} = LS_{\mathbf{a},\mathbf{b}} = 8$

5. Since $\hat{\tau} < \min(LS_{\mathbf{a}}, LS_{\mathbf{b}})$ we can create a cut of the form:

   $w_{\mathbf{a},\hat{\tau}} + w_{\mathbf{b},\hat{\tau}} \geq 1 \qquad \Rightarrow \qquad w_{10,8} + w_{12,8} \geq 1$

This means that by the end of time period 8, at least one of the two blocks represented by block 10 and block 12 in the figure above must be mined.

4.3.4 Three-Way Earliest Starts Cuts

Generating cuts with three blocks is significantly more complicated than generating cuts with just two blocks. With three blocks there are many more combinations to investigate. Also, the right-hand-side of the constraint can assume two different values, either 1 or 2. Therefore, the resultant cuts can take either of the following two forms:

$$w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1 \quad \text{or} \quad w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$$

where **a**, **b**, and **c** are arbitrarily chosen blocks that adhere to the *reasonable block selection rule* (see Section 4.3.1). The first cut allows at most one of the three blocks to be mined by a particular time period, while the second cut allows at most two of three blocks to be mined by a particular time period. As with two-way earliest starts cuts, we employ the maximum production and/or maximum processing constraints and the support weights of various blocks to construct our cuts.

Let us assume that we can access all three blocks (**a**, **b**, and **c**) by time period $\bar{\tau}$ (i.e., $ES_{\mathbf{a},\mathbf{b},\mathbf{c}} = \bar{\tau}$). Additionally, let us assume that the *earliest* earliest start time for all the two-way combinations is $\bar{\tau}'$ (i.e., $\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right)$). First, we need to determine how many blocks are accessible before $\bar{\tau}$. If the *earliest* earliest start time for all two-way combinations of the three blocks is less than $\bar{\tau}$ (i.e., $\bar{\tau}' < \bar{\tau}$), then by time period $(\bar{\tau} - 1)$ at most two of the three blocks are accessible and it is valid to limit access to at most two of these three blocks. Next, we need to determine how many blocks are accessible before $\bar{\tau}'$. If the *earliest* single block earliest start time for all three blocks is less than $\bar{\tau}'$ (i.e., $\min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right) < \bar{\tau}'$), then by time period $(\bar{\tau}' - 1)$ at most one of the three blocks is accessible and it is valid to limit access to at most one of these three blocks.

More specifically, we use the earliest starts algorithm to determine all single block, pair-wise, and three-way block combination earliest starts (ES) for our three

blocks and then define the following:

- $\bar{\tau}'' = \min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right)$
- $\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right)$
- $\bar{\tau} = ES_{\mathbf{a},\mathbf{b},\mathbf{c}}$

We then use these three values ($\bar{\tau}$, $\bar{\tau}'$, and $\bar{\tau}''$) to generate our cuts.

If $\bar{\tau}$ is greater than $\bar{\tau}'$, then the blocks that comprise the super-block formed by the union of blocks **a**, **b**, and **c** can only be accessed by a time period later than the earliest start times for any of the super-blocks formed by two-way combinations of blocks **a**, **b**, and **c**. As a result, access is limited to only two of the these three blocks by time period $\bar{\tau} - 1$ and an appropriate cut of the form $w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$ can be generated.

If $\bar{\tau}'$ is greater than $\bar{\tau}''$, then any two blocks that comprise the super-block formed by the union of blocks **a**, **b**, and **c** can only be accessed by a time period later than the earliest start times for any of the blocks **a**, **b**, and **c** individually. As a result, access is limited to only one of the these three blocks by time period $\bar{\tau}' - 1$ and an appropriate cut of the form $w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$ can be generated.

Determining if the Three-Way Earliest Starts Cuts are Valid and Useful As with two-way earliest starts cuts, only those three-way cuts that are valid and useful should be included in the model formulation. We must ensure that both types of cuts we generate (≤ 2 and ≤ 1) are valid and useful.

To determine if our cuts of the form $w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$ meet these criteria, we must pay particular attention to the time index $\bar{\tau} - 1$. Based on the earliest start of the super-block formed by the union of blocks **a**, **b**, and **c** ($ES_{\mathbf{a},\mathbf{b},\mathbf{c}}$, which we call $\bar{\tau}$), we know that $\bar{\tau}$ is the earliest possible time period that all three of these blocks can be mined together. This means that during any time period before

$\bar{\tau}$, at most two of these three blocks can be mined. So it is valid to limit access to at most two of these three blocks by time period $\bar{\tau} - 1$.

It may be *practically* useful to limit at most two of these three blocks **a**, **b**, and **c** to be accessed by time period $\bar{\tau} - 1$, because if the values of two of the blocks are known to be *mined* (i.e., say $w_{\mathbf{a},\bar{\tau}-1} = w_{\mathbf{b},\bar{\tau}-1} = 1$) then the value of the other block is also known due to the cut ($w_{\mathbf{c},\bar{\tau}-1}$ must equal 0 or the constraint represented by the cut is violated). To determine the *theoretical* usefulness of the cut, however, we must empirically test each cut with specific data. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For three-way earliest starts cuts of the form $w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\bar{\tau}-1}$, $w_{\mathbf{b},\bar{\tau}-1}$, and $w_{\mathbf{c},\bar{\tau}-1}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\bar{\tau}-1}$, $\tilde{w}_{\mathbf{b},\bar{\tau}-1}$, and $\tilde{w}_{\mathbf{c},\bar{\tau}-1}$) is greater than 2:

$$\tilde{w}_{\mathbf{a},\bar{\tau}-1} + \tilde{w}_{\mathbf{b},\bar{\tau}-1} + \tilde{w}_{\mathbf{c},\bar{\tau}-1} > 2$$

To determine if our cuts of the form $w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$ are valid and useful, we must pay particular attention to the time index $\bar{\tau}'-1$. Based on the *earliest* earliest start of the super-block formed by the union of any two of the blocks **a**, **b**, and **c** ($\min(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}})$, which we call $\bar{\tau}'$), we know that $\bar{\tau}'$ is the earliest possible time period that any two of the three blocks can be mined together. This means that during any time period before $\bar{\tau}'$, at most one of these three blocks can be mined. So it is valid to limit access to at most one of these three blocks by time period $\bar{\tau}' - 1$.

It may be *practically* useful to limit at most one of these three blocks **a**, **b**, and **c** to be accessed by time period $\bar{\tau}' - 1$, because if the value of one of the blocks is known to be *mined* (i.e., say $w_{\mathbf{a},\bar{\tau}'-1} = 1$) then the values of the other two blocks are also known due to the cut ($w_{\mathbf{b},\bar{\tau}'-1} = w_{\mathbf{c},\bar{\tau}'-1} = 0$ or the constraint represented

by the cut is violated). To determine the *theoretical* usefulness of the cut, however, we must again resort to empirical tests. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For three-way earliest starts cuts of the form $w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\bar{\tau}'-1}$, $w_{\mathbf{b},\bar{\tau}'-1}$, and $w_{\mathbf{c},\bar{\tau}'-1}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\bar{\tau}'-1}$, $\tilde{w}_{\mathbf{b},\bar{\tau}'-1}$, and $\tilde{w}_{\mathbf{c},\bar{\tau}'-1}$) is greater than 1:

$$\tilde{w}_{\mathbf{a},\bar{\tau}'-1} + \tilde{w}_{\mathbf{b},\bar{\tau}'-1} + \tilde{w}_{\mathbf{c},\bar{\tau}'-1} > 1$$

It is interesting to note that if $\bar{\tau}' \not> \bar{\tau}''$ (which implies that $\bar{\tau}' = \bar{\tau}''$ because $\bar{\tau}' < \bar{\tau}''$ is impossible), then although the cut is valid (no optimal answers are precluded from being examined), it is not useful (practically or theoretically). The reason it is not useful is because $\bar{\tau}''$ tells us the earliest start time that any single block can be accessed, so if $\bar{\tau}' = \bar{\tau}''$, then by time $\bar{\tau}' - 1$ none of the single blocks will be accessible due to their single block earliest start times. Essentially, this cut just tells us something we already know because of each individual block's earliest start time.

Three-Way Earliest Starts Cuts Algorithm

Assumptions We again use all assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- $\mathbf{a}$ = A block (from the set of blocks B) which adheres to the *reasonable block selection rule*
- $\mathbf{b}$ = Another block (from the set of blocks B and not the same as $\mathbf{a}$) which adheres to the *reasonable block selection rule*

- $\mathbf{c}$ = Another block (from the set of blocks B and not the same as $\mathbf{a}$ or $\mathbf{b}$) which adheres to the *reasonable block selection rule*

- $S_{\mathbf{a},\mathbf{b}}$ = Set of blocks that must be mined (based on the sequencing constraints) in order to mine blocks $\mathbf{a}$ and $\mathbf{b}$ (including explicitly mining blocks $\mathbf{a}$ and $\mathbf{b}$). This set contains blocks $\mathbf{a}$ and $\mathbf{b}$ and the union of all the blocks in each of their respective precedence sets (i.e., $S_{\mathbf{a},\mathbf{b}} = S_{\mathbf{a}} \cup S_{\mathbf{b}}$, since $S_{\mathbf{a}}$ contains block $\mathbf{a}$ and all the blocks in block $\mathbf{a}$'s precedence set and $S_{\mathbf{b}}$ contains block $\mathbf{b}$ and all the blocks in block $\mathbf{b}$'s precedence set). As a result, no shared blocks between the precedence sets of blocks $\mathbf{a}$ and $\mathbf{b}$ are counted more than once in the super-block represented by $S_{\mathbf{a},\mathbf{b}}$.

- $S_{\mathbf{a},\mathbf{c}}$ = Set of blocks that must be mined (based on the sequencing constraints) in order to mine blocks $\mathbf{a}$ and $\mathbf{c}$ (including explicitly mining blocks $\mathbf{a}$ and $\mathbf{c}$). This set contains blocks $\mathbf{a}$ and $\mathbf{c}$ and the union of all the blocks in each of their respective precedence sets (i.e., $S_{\mathbf{a},\mathbf{c}} = S_{\mathbf{a}} \cup S_{\mathbf{c}}$, since $S_{\mathbf{a}}$ contains block $\mathbf{a}$ and all the blocks in block $\mathbf{a}$'s precedence set and $S_{\mathbf{c}}$ contains block $\mathbf{c}$ and all the blocks in block $\mathbf{c}$'s precedence set). As a result, no shared blocks between the precedence sets of blocks $\mathbf{a}$ and $\mathbf{c}$ are counted more than once in the super-block represented by $S_{\mathbf{a},\mathbf{c}}$.

- $S_{\mathbf{b},\mathbf{c}}$ = Set of blocks that must be mined (based on the sequencing constraints) in order to mine blocks $\mathbf{b}$ and $\mathbf{c}$ (including explicitly mining blocks $\mathbf{b}$ and $\mathbf{c}$). This set contains blocks $\mathbf{b}$ and $\mathbf{c}$ and the union of all the blocks in each of their respective precedence sets (i.e., $S_{\mathbf{b},\mathbf{c}} = S_{\mathbf{b}} \cup S_{\mathbf{c}}$, since $S_{\mathbf{b}}$ contains block $\mathbf{b}$ and all the blocks in block $\mathbf{b}$'s precedence set and $S_{\mathbf{c}}$ contains block $\mathbf{c}$ and all the blocks in block $\mathbf{c}$'s precedence set). As a result, no shared blocks between the precedence sets of blocks $\mathbf{b}$ and $\mathbf{c}$ are counted more than once in the super-block represented by $S_{\mathbf{b},\mathbf{c}}$.

- $S_{\mathbf{a},\mathbf{b},\mathbf{c}}$ = Set of blocks that must be mined (based on the sequencing constraints) in order to mine blocks **a**, **b**, and **c** (including explicitly mining blocks **a**, **b** and **c**). This set contains blocks **a**, **b**, and **c** and the union of all the blocks in each of their respective precedence sets (i.e., $S_{\mathbf{a},\mathbf{b},\mathbf{c}} = S_{\mathbf{a}} \cup S_{\mathbf{b}} \cup S_{\mathbf{c}}$, since $S_{\mathbf{a}}$ contains block **a** and all the blocks in block **a**'s precedence set, $S_{\mathbf{b}}$ contains block **b** and all the blocks in block **b**'s precedence set, and $S_{\mathbf{c}}$ contains block **c** and all the blocks in block **c**'s precedence set). As a result, no shared blocks between the precedence sets of blocks **a**, **b**, and **c** are counted more than once in the super-block represented by $S_{\mathbf{a},\mathbf{b},\mathbf{c}}$.

- $ES_{\mathbf{a}}$ = Earliest start time for block **a** (based on either the maximum processing constraint or the maximum production constraint)

- $ES_{\mathbf{b}}$ = Earliest start time for block **b** (based on either the maximum processing constraint or the maximum production constraint)

- $ES_{\mathbf{c}}$ = Earliest start time for block **c** (based on either the maximum processing constraint or the maximum production constraint)

- $\bar{\tau}'' = \min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right)$

- $ES_{\mathbf{a},\mathbf{b}}$ = Earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b}}$ (based on either the maximum processing constraint or the maximum production constraint)

- $ES_{\mathbf{a},\mathbf{c}}$ = Earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{c}}$ (based on either the maximum processing constraint or the maximum production constraint)

- $ES_{\mathbf{b},\mathbf{c}}$ = Earliest start time for the set of blocks contained in $S_{\mathbf{b},\mathbf{c}}$ (based on either the maximum processing constraint or the maximum production constraint)

- $\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right)$

- $\bar{\tau} = ES_{\mathbf{a},\mathbf{b},\mathbf{c}}$ = Earliest start time for the set of blocks contained in $S_{\mathbf{a},\mathbf{b},\mathbf{c}}$ (based on either the maximum processing constraint or the maximum production constraint)

Inputs

- A set of blocks B
- Maximum processing capacity per time period (in tons of ore) and maximum production capacity per time period (in tons of material). Note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- Valid cuts of the form:

$$w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$$

and

$$w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$$

Algorithm

For each three-way combination of blocks $\mathbf{a} \in B$, $\mathbf{b} \in B$, and $\mathbf{c} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the maximum *production* capacity **do**:

1a. Determine the earliest start time for block $\mathbf{a}$ (i.e., $ES_{\mathbf{a}}$) based on the maximum production capacity

2a. Determine the earliest start time for block $\mathbf{b}$ (i.e., $ES_{\mathbf{b}}$) based on the maximum production capacity

3a. Determine the earliest start time for block **c** (i.e., $ES_{\mathbf{c}}$) based on the maximum production capacity

4a. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **b** (i.e., $S_{\mathbf{a},\mathbf{b}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{a},\mathbf{b}}$) based on the maximum production capacity

5a. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **c** (i.e., $S_{\mathbf{a},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{a},\mathbf{c}}$) based on the maximum production capacity

6a. Create the set of blocks that represents the union of the precedence sets for blocks **b** and **c** (i.e., $S_{\mathbf{b},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{b},\mathbf{c}}$) based on the maximum production capacity

7a. Create the set of blocks that represents the union of the precedence sets for blocks **a**, **b**, and **c** (i.e., $S_{\mathbf{a},\mathbf{b},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $\bar{\tau} = ES_{\mathbf{a},\mathbf{b},\mathbf{c}}$) based on the maximum production capacity

8a. Determine the earliest that any two-block set can be accessed:

$$\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right)$$

9a. Determine the earliest that any single block can be accessed:

$$\bar{\tau}'' = \min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right)$$

10a. If $\bar{\tau} > \bar{\tau}'$ then create a cut of the form:

$$w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$$

11a. If $\bar{\tau}' > \bar{\tau}''$ then create a cut of the form:

$$w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$$

For each three-way combination of blocks $\mathbf{a} \in B$, $\mathbf{b} \in B$, and $\mathbf{c} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the maximum *processing* capacity **do**:

1b. Determine the earliest start time for block $\mathbf{a}$ (i.e., $ES_{\mathbf{a}}$) based on the maximum processing capacity

2b. Determine the earliest start time for block $\mathbf{b}$ (i.e., $ES_{\mathbf{b}}$) based on the maximum processing capacity

3b. Determine the earliest start time for block $\mathbf{c}$ (i.e., $ES_{\mathbf{c}}$) based on the maximum processing capacity

4b. Create the set of blocks that represents the union of the precedence sets for blocks $\mathbf{a}$ and $\mathbf{b}$ (i.e., $S_{\mathbf{a},\mathbf{b}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{a},\mathbf{b}}$) based on the maximum processing capacity

5b. Create the set of blocks that represents the union of the precedence sets for blocks $\mathbf{a}$ and $\mathbf{c}$ (i.e., $S_{\mathbf{a},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{a},\mathbf{c}}$) based on the maximum processing capacity

6b. Create the set of blocks that represents the union of the precedence sets for blocks $\mathbf{b}$ and $\mathbf{c}$ (i.e., $S_{\mathbf{b},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $ES_{\mathbf{b},\mathbf{c}}$) based on the maximum processing capacity

7b. Create the set of blocks that represents the union of the precedence sets for blocks $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ (i.e., $S_{\mathbf{a},\mathbf{b},\mathbf{c}}$) and determine the earliest start time for this set (i.e., $\bar{\tau} = ES_{\mathbf{a},\mathbf{b},\mathbf{c}}$) based on the maximum processing capacity

8b. Determine the earliest that any two-block set can be accessed:

$$\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right)$$

9b. Determine the earliest that any single block can be accessed:

$$\bar{\tau}'' = \min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right)$$

10b. If $\bar{\tau} > \bar{\tau}'$ then create a cut of the form:

$$w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2$$

11b. If $\bar{\tau}' > \bar{\tau}''$ then create a cut of the form:

$$w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1$$

Output all generated cuts

Relative Dominance of Three-Way Earliest Starts Cuts As with two-way cuts, the time index ($\bar{\tau}$ or $\bar{\tau}'$) in our generated cuts tells us about the relative dominance of different cuts. Again, cuts with a later time index dominate cuts (involving the same blocks) with an earlier time index.

Three-way cuts concern themselves with another dominance issue though; the value to the right of the inequality. Take the following two potential cuts:

- $w_{\mathbf{a},2} + w_{\mathbf{b},2} + w_{\mathbf{c},2} \leq 1$
- $w_{\mathbf{a},2} + w_{\mathbf{b},2} + w_{\mathbf{c},2} \leq 2$

The first means that at most one of the three blocks **a**, **b**, and **c** can be mined by time period 2, while the second means that at most two of these three blocks can be mined

by time period 2. In this case, the former cut dominates the latter. The reason for this is that the former is more restrictive than the latter. The former restricts access to only one block, while the latter allows access to any two of the blocks. Our three-way earliest starts cuts algorithm accounts for this dominance and only generates the dominant cut for any set of blocks in the same time period (assuming such a cut is valid and useful).

Three-Way Earliest Starts Cuts Numerical Example Looking at our two-dimensional example, we use blocks 9, 11, and 13 to create a three-way earliest starts cut based on the maximum production capacity (see Figure 4.6).

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.6. Three-Way Earliest Starts Cuts Numerical Example. This example depicts the results of creating a three-way earliest start cut with blocks 9, 11, and 13.

For this example, we assume that:

- Block **a** is represented by block 9 in the figure, block **b** is represented by block 11 in the figure, and block **c** is represented by block 13 in the figure
- Each block contains 10 tons of material (i.e., $n_b = 10$ for each block **a**, **b**, and **c**)
- The maximum production capacity is 40 tons per time period (for simplicity, we only use production bounds for this example)

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block (blocks 9, 11, and 13 in the figure above for this example) and the t index identifies the time period by which the block is extracted. We now use the algorithm to generate a three-way earliest starts cut based on the maximum production capacity:

1. Determine the earliest start time for block **a**:

   $ES_{\mathbf{a}} = 1$

2. Determine the earliest start time for block **b:**

   $ES_{\mathbf{b}} = 1$

3. Determine the earliest start time for block **c:**

   $ES_{\mathbf{c}} = 1$

4. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **b** and determine this set's earliest start time:

   $S_{\mathbf{a},\mathbf{b}} = S_{\mathbf{a}} \cup S_{\mathbf{b}} = \{1, 2, 3, 9\} \cup \{3, 4, 5, 11\} = \{1, 2, 3, 4, 5, 9, 11\}$

   $ES_{\mathbf{a},\mathbf{b}} = 2$

5. Create the set of blocks that represents the union of the precedence sets for blocks **a** and **c** and determine this set's earliest start time:

   $S_{\mathbf{a},\mathbf{c}} = S_{\mathbf{a}} \cup S_{\mathbf{c}} = \{1, 2, 3, 9\} \cup \{5, 6, 7, 13\} = \{1, 2, 3, 5, 6, 7, 9, 13\}$

   $ES_{\mathbf{a},\mathbf{c}} = 2$

6. Create the set of blocks that represents the union of the precedence sets for blocks **b** and **c** and determine this set's earliest start time:

   $S_{\mathbf{b},\mathbf{c}} = S_{\mathbf{b}} \cup S_{\mathbf{c}} = \{3, 4, 5, 11\} \cup \{5, 6, 7, 13\} = \{3, 4, 5, 6, 7, 11, 13\}$

   $ES_{\mathbf{b},\mathbf{c}} = 2$

7. Create the set of blocks that represents the union of the precedence sets for blocks **a**, **b**, and **c** and determine this set's earliest start time:

   $S_{\mathbf{a},\mathbf{b},\mathbf{c}} = S_{\mathbf{a}} \cup S_{\mathbf{b}} \cup S_{\mathbf{c}} = \{1,2,3,9\} \cup \{3,4,5,11\} \cup \{5,6,7,13\}$

   $= \{1,2,3,4,5,6,7,9,11,13\}$

   $\bar{\tau} = ES_{\mathbf{a},\mathbf{b},\mathbf{c}} = 3$

8. Determine the earliest that any two-block set can be accessed:

   $\bar{\tau}' = \min\left(ES_{\mathbf{a},\mathbf{b}}, ES_{\mathbf{a},\mathbf{c}}, ES_{\mathbf{b},\mathbf{c}}\right) = \min(2,2,2) = 2$

9. Determine the earliest that any single block can be accessed:

   $\bar{\tau}'' = \min\left(ES_{\mathbf{a}}, ES_{\mathbf{b}}, ES_{\mathbf{c}}\right) = \min(1,1,1) = 1$

10. Since $\bar{\tau} > \bar{\tau}'$ we can create a cut of the form:

    $w_{\mathbf{a},\bar{\tau}-1} + w_{\mathbf{b},\bar{\tau}-1} + w_{\mathbf{c},\bar{\tau}-1} \leq 2 \qquad \Rightarrow \qquad w_{9,2} + w_{11,2} + w_{13,2} \leq 2$

11. Since $\bar{\tau}' > \bar{\tau}''$ we can create a cut of the form:

    $w_{\mathbf{a},\bar{\tau}'-1} + w_{\mathbf{b},\bar{\tau}'-1} + w_{\mathbf{c},\bar{\tau}'-1} \leq 1 \qquad \Rightarrow \qquad w_{9,1} + w_{11,1} + w_{13,1} \leq 1$

This means that by the end of time period 2, at most two of the three blocks represented by blocks 9, 11, and 13 in the figure above can be mined. Also, by the end of time period 1, at most one of the three blocks represented by blocks 9, 11, and 13 in the figure above can be mined.

4.3.5 Three-Way Latest Starts Cuts

As is the case in generating three-way earliest starts cuts, three-way latest starts cuts can assume either of the following two forms:

$$w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1 \quad \text{or} \quad w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$$

where **a**, **b**, and **c** are arbitrarily chosen blocks that adhere to the *reasonable block selection rule* (see Section 4.3.1). The first cut requires that at least one of three blocks be mined by a particular time period, while the second cut requires that at least two of three blocks be mined by a particular time period. As with two-way latest starts cuts, we employ the minimum production and/or minimum processing constraints and the holding weights of various blocks to construct our cuts.

Let us assume that we must mine all three blocks (**a**, **b**, and **c**) by time period $\tilde{\tau}$ (i.e., $LS_{\mathbf{a},\mathbf{b},\mathbf{c}} = \tilde{\tau}$). Additionally, let us assume that the *latest* latest start time for all the two-way combinations is $\tilde{\tau}'$ (i.e., $\tilde{\tau}' = \max(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}})$). First, we need to determine how many blocks must be mined after $\tilde{\tau}$. If the *latest* latest start time for all two-way combinations of the three blocks is later than $\tilde{\tau}$ (i.e., $\tilde{\tau}' > \tilde{\tau}$), then by time period $\tilde{\tau}$ at least one of the three blocks must be mined and it is valid to force at least one of the decision variables representing these three blocks to assume a value of 1 (i.e., *mined*). Next, we need to determine how many blocks must start to be mined after $\tilde{\tau}'$. If the *latest* single block latest start time for all three blocks is later than $\tilde{\tau}'$ (i.e., $\max(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}}) > \tilde{\tau}'$), then by time period $\tilde{\tau}'$ at least two of the three blocks must be mined and it is valid to force at least two of the decision variables representing these three blocks to assume a value of 1 (i.e., *mined*).

More specifically, we use the latest starts algorithm to determine all single block, pair-wise, and three-way block combination latest starts (LS) for our three blocks and then define the following:

- $\tilde{\tau}'' = \max(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}})$
- $\tilde{\tau}' = \max(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}})$
- $\tilde{\tau} = LS_{\mathbf{a},\mathbf{b},\mathbf{c}}$

We then use these three values ($\tilde{\tau}$, $\tilde{\tau}'$, and $\tilde{\tau}''$) to generate our cuts.

If $\tilde{\tau}$ is less than $\tilde{\tau}'$, then the blocks that comprise the super-block formed by the union of blocks **a**, **b**, and **c** must be accessed by a time period earlier than the latest

start times for any of the super-blocks formed by two-way combinations of blocks **a**, **b**, and **c**. As a result, at least one of these three blocks must be removed by time period $\tilde{\tau}$ and an appropriate cut of the form $w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$ can be generated.

If $\tilde{\tau}'$ is less than $\tilde{\tau}''$, then any two blocks that comprise the super-block formed by the union of blocks **a**, **b**, and **c** must be accessed by a time period earlier than the latest start times for any of the blocks **a**, **b**, and **c** individually. As a result, at least two of these three blocks must be removed by time period $\tilde{\tau}'$ and an appropriate cut of the form $w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$ can be generated.

Determining if the Three-Way Latest Starts Cuts are Valid and Useful As with two-way latest starts cuts, only those three-way cuts that are valid and useful should be included in the model formulation. We must ensure that both types of cuts we generate (≥ 2 and ≥ 1) are valid and useful.

To determine if our cuts of the form $w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$ meet these criteria, we must pay particular attention to the time index $\tilde{\tau}$. Based on the latest start of the super-block formed by the union of blocks **a**, **b**, and **c** ($LS_{\mathbf{a},\mathbf{b},\mathbf{c}}$, which we call $\tilde{\tau}$), we know that by time period $\tilde{\tau}$ all three blocks **a**, **b**, and **c** are holding up access to the remaining blocks in the pit. Even if there exists a two-way combination of blocks **a**, **b**, and **c** that does not need to be accessed until a later time period (i.e., its two-way latest start is later than $\tilde{\tau}$), then we still need to remove at least one block during time period $\tilde{\tau}$ to meet the minimum production and/or processing requirements. This means that by time period $\tilde{\tau}$, at least one of these three blocks must be mined. So it is valid to force at least one of these three blocks to be mined by time period $\tilde{\tau}$.

It may be *practically* useful to force at least one of these three blocks **a**, **b**, and **c** to be accessed by time period $\tilde{\tau}$, because if the values of two of the blocks are known to be *not mined* (i.e., say $w_{\mathbf{a},\tilde{\tau}} = w_{\mathbf{b},\tilde{\tau}} = 0$) then the value of the other block is also known due to the cut ($w_{\mathbf{c},\tilde{\tau}}$ must equal 1 or the constraint represented by the cut is violated). To determine the *theoretical* usefulness of the cut, however, we must

empirically test each cut with specific data. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For three-way latest starts cuts of the form $w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\tilde{\tau}}$, $w_{\mathbf{b},\tilde{\tau}}$, and $w_{\mathbf{c},\tilde{\tau}}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\tilde{\tau}}$, $\tilde{w}_{\mathbf{b},\tilde{\tau}}$, and $\tilde{w}_{\mathbf{c},\tilde{\tau}}$) is less than 1:

$$\tilde{w}_{\mathbf{a},\tilde{\tau}} + \tilde{w}_{\mathbf{b},\tilde{\tau}} + \tilde{w}_{\mathbf{c},\tilde{\tau}} < 1$$

To determine if our cuts of the form $w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$ are valid and useful, we must pay particular attention to the time index $\tilde{\tau}'$. Based on the *latest* latest start of the super-block formed by the union of any two of the blocks **a**, **b**, and **c** ($\max(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}})$, which we call $\tilde{\tau}'$), we know that by time period $\tilde{\tau}'$ at least two of the three blocks **a**, **b**, and **c** are holding up access to the remaining blocks in the pit. Even if there exists a block among **a**, **b**, and **c** that does not need to be accessed until a later time period (i.e., its latest start is later than $\tilde{\tau}'$), then we still need to remove at least two blocks during time period $\tilde{\tau}'$ to meet the minimum production and/or processing requirements. This means that by time period $\tilde{\tau}'$, at least two of these three blocks **a**, **b**, and **c** must be mined. So it is valid to force at least two of these three blocks to be mined by time period $\tilde{\tau}'$.

It may be *practically* useful to force at least two of these three blocks **a**, **b**, and **c** to be accessed by time period $\tilde{\tau}'$, because if the value of one of the blocks is known to be *not mined* (i.e., say $w_{\mathbf{a},\tilde{\tau}'} = 0$) then the values of the other two blocks are also known due to the cut ($w_{\mathbf{b},\tilde{\tau}'} = w_{\mathbf{c},\tilde{\tau}'} = 1$ or the constraint represented by the cut is violated). To determine the *theoretical* usefulness of the cut, however, we must again resort to empirical tests. We consider a cut theoretically useful if, among other things, it renders infeasible the optimal solution to the LP relaxation of the original integer programming formulation. For three-way latest starts cuts of the form $w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$, the cut is useful if the sum of the values of the variables $w_{\mathbf{a},\tilde{\tau}'}$,

$w_{\mathbf{b},\tilde{\tau}'}$, and $w_{\mathbf{c},\tilde{\tau}'}$ in the optimal LP relaxation (we call them $\tilde{w}_{\mathbf{a},\tilde{\tau}'}$, $\tilde{w}_{\mathbf{b},\tilde{\tau}'}$, and $\tilde{w}_{\mathbf{c},\tilde{\tau}'}$) is less than 2:

$$\tilde{w}_{\mathbf{a},\tilde{\tau}'} + \tilde{w}_{\mathbf{b},\tilde{\tau}'} + \tilde{w}_{\mathbf{c},\tilde{\tau}'} < 2$$

It is interesting to note that if $\tilde{\tau}' \not< \tilde{\tau}''$ (which implies that $\tilde{\tau}' = \tilde{\tau}''$ because $\tilde{\tau}' > \tilde{\tau}''$ is impossible), then although the cut is valid (no optimal answers are precluded from being examined), it is not useful (practically or theoretically). The reason it is not useful is because $\tilde{\tau}''$ tells us the latest start time that any single block must be accessed, so if $\tilde{\tau}' = \tilde{\tau}''$, then by time $\tilde{\tau}'$ all of the single blocks must be mined due to their single block latest start times. Essentially, this cut just tells us something we already know because of each individual block's latest start time.

Three-Way Latest Starts Cuts Algorithm

Assumptions We again use all assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- $\mathbf{a}$ = A block (from the set of blocks B) which adheres to the *reasonable block selection rule*
- $\mathbf{b}$ = Another block (from the set of blocks B and not the same as $\mathbf{a}$) which adheres to the *reasonable block selection rule*
- $\mathbf{c}$ = Another block (from the set of blocks B and not the same as $\mathbf{a}$ or $\mathbf{b}$) which adheres to the *reasonable block selection rule*
- $H_{\mathbf{a},\mathbf{b}}$ = Set of blocks that cannot be mined (based on the sequencing constraints) until blocks $\mathbf{a}$ and $\mathbf{b}$ are mined (including explicitly mining blocks $\mathbf{a}$ and $\mathbf{b}$). This set contains blocks $\mathbf{a}$ and $\mathbf{b}$ and the union of all the blocks in each of their respective holding sets (i.e., $H_{\mathbf{a},\mathbf{b}} = H_{\mathbf{a}} \cup H_{\mathbf{b}}$, since $H_{\mathbf{a}}$ contains block $\mathbf{a}$ and all

the blocks in block **a**'s holding set and $H_{\mathbf{b}}$ contains block **b** and all the blocks in block **b**'s holding set). As a result, no shared blocks between the holding sets of blocks **a** and **b** are counted more than once in the super-block represented by $H_{\mathbf{a,b}}$.

- $H_{\mathbf{a,c}}$ = Set of blocks that cannot be mined (based on the sequencing constraints) until blocks **a** and **c** are mined (including explicitly mining blocks **a** and c). This set contains blocks **a** and **c** and the union of all the blocks in each of their respective holding sets (i.e., $H_{\mathbf{a,c}} = H_{\mathbf{a}} \cup H_{\mathbf{c}}$, since $H_{\mathbf{a}}$ contains block **a** and all the blocks in block **a**'s holding set and $H_{\mathbf{c}}$ contains block **c** and all the blocks in block **c**'s holding set). As a result, no shared blocks between the holding sets of blocks **a** and **c** are counted more than once in the super-block represented by $H_{\mathbf{a,c}}$.

- $H_{\mathbf{b,c}}$ = Set of blocks that cannot be mined (based on the sequencing constraints) until blocks **b** and **c** are mined (including explicitly mining blocks **b** and **c**). This set contains blocks **b** and **c** and the union of all the blocks in each of their respective holding sets (i.e., $H_{\mathbf{b,c}} = H_{\mathbf{b}} \cup H_{\mathbf{c}}$, since $H_{\mathbf{b}}$ contains block **b** and all the blocks in block **b**'s holding set and $H_{\mathbf{c}}$ contains block **c** and all the blocks in block **c**'s holding set). As a result, no shared blocks between the holding sets of blocks **b** and **c** are counted more than once in the super-block represented by $H_{\mathbf{b,c}}$.

- $H_{\mathbf{a,b,c}}$ = Set of blocks that cannot be mined (based on the sequencing constraints) until blocks **a**, **b**, and **c** are mined (including explicitly mining blocks **a**, **b**, and **c**). This set contains blocks **a**, **b**, and **c** and the union of all the blocks in each of their respective holding sets (i.e., $H_{\mathbf{a,b,c}} = H_{\mathbf{a}} \cup H_{\mathbf{b}} \cup H_{\mathbf{c}}$, since $H_{\mathbf{a}}$ contains block **a** and all the blocks in block **a**'s holding set, $H_{\mathbf{b}}$ contains block **b** and all the blocks in block **b**'s holding set, and $H_{\mathbf{c}}$ contains block **c** and all the blocks in block **c**'s holding set). As a result, no shared blocks between the

holding sets of blocks **a**, **b**, and **c** are counted more than once in the super-block represented by $H_{\mathbf{a},\mathbf{b},\mathbf{c}}$.

- $LS_{\mathbf{a}}$ = Latest start time for block **a** (based on either the minimum processing constraint or the minimum production constraint)
- $LS_{\mathbf{b}}$ = Latest start time for block **b** (based on either the minimum processing constraint or the minimum production constraint)
- $LS_{\mathbf{c}}$ = Latest start time for block **c** (based on either the minimum processing constraint or the minimum production constraint)
- $\tilde{\tau}'' = \max\left(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}}\right)$
- $LS_{\mathbf{a},\mathbf{b}}$ = Latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b}}$ (based on either the minimum processing constraint or the minimum production constraint)
- $LS_{\mathbf{a},\mathbf{c}}$ = Latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{c}}$ (based on either the minimum processing constraint or the minimum production constraint)
- $LS_{\mathbf{b},\mathbf{c}}$ = Latest start time for the set of blocks contained in $H_{\mathbf{b},\mathbf{c}}$ (based on either the minimum processing constraint or the minimum production constraint)
- $\tilde{\tau}' = \max\left(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}}\right)$
- $\tilde{\tau} = LS_{\mathbf{a},\mathbf{b},\mathbf{c}}$ = Latest start time for the set of blocks contained in $H_{\mathbf{a},\mathbf{b},\mathbf{c}}$ (based on either the minimum processing constraint or the minimum production constraint)

Inputs

- A set of blocks B

- Minimum processing requirement per time period (in tons of ore) and minimum production requirement per time period (in tons of material). Note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized).

Outputs

- Valid cuts of the form:

$$w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$$

and

$$w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$$

Algorithm

For each three-way combination of blocks $\mathbf{a} \in B$, $\mathbf{b} \in B$, and $\mathbf{c} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the minimum *production* requirement **do**:

1a. Determine the latest start time for block $\mathbf{a}$ (i.e., $LS_{\mathbf{a}}$) based on the minimum production capacity

2a. Determine the latest start time for block $\mathbf{b}$ (i.e., $LS_{\mathbf{b}}$) based on the minimum production capacity

3a. Determine the latest start time for block $\mathbf{c}$ (i.e., $LS_{\mathbf{c}}$) based on the minimum production capacity

4a. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{a}$ and $\mathbf{b}$ (i.e., $H_{\mathbf{a},\mathbf{b}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{a},\mathbf{b}}$) based on the minimum production capacity

5a. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{a}$ and $\mathbf{c}$ (i.e., $H_{\mathbf{a},\mathbf{c}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{a},\mathbf{c}}$) based on the minimum production capacity

6a. Create the set of blocks that represents the union of the holding sets for blocks **b** and **c** (i.e., $H_{\mathbf{b},\mathbf{c}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{b},\mathbf{c}}$) based on the minimum production capacity

7a. Create the set of blocks that represents the union of the holding sets for blocks **a**, **b**, and **c** (i.e., $H_{\mathbf{a},\mathbf{b},\mathbf{c}}$) and determine the latest start time for this set (i.e., $\tilde{\tau} = LS_{\mathbf{a},\mathbf{b},\mathbf{c}}$) based on the minimum production capacity

8a. Determine the latest that any two-block set must be accessed:

$$\tilde{\tau}' = \max\left(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}}\right)$$

9a. Determine the latest that any single block must be accessed:

$$\tilde{\tau}'' = \max\left(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}}\right)$$

10a. If $\tilde{\tau} < \tilde{\tau}'$ then create a cut of the form:

$$w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$$

11a. If $\tilde{\tau}' < \tilde{\tau}''$ then create a cut of the form:

$$w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$$

For each three-way combination of blocks $\mathbf{a} \in B$, $\mathbf{b} \in B$, and $\mathbf{c} \in B$ in which each block adheres to the *reasonable block selection rule* with respect to the minimum *processing* requirement **do**:

1b. Determine the latest start time for block **a** (i.e., $LS_{\mathbf{a}}$) based on the minimum processing capacity

2b. Determine the latest start time for block **b** (i.e., $LS_{\mathbf{b}}$) based on the mini-

mum processing capacity

3b. Determine the latest start time for block $\mathbf{c}$ (i.e., $LS_{\mathbf{c}}$) based on the minimum processing capacity

4b. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{a}$ and $\mathbf{b}$ (i.e., $H_{\mathbf{a},\mathbf{b}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{a},\mathbf{b}}$) based on the minimum processing capacity

5b. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{a}$ and $\mathbf{c}$ (i.e., $H_{\mathbf{a},\mathbf{c}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{a},\mathbf{c}}$) based on the minimum processing capacity

6b. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{b}$ and $\mathbf{c}$ (i.e., $H_{\mathbf{b},\mathbf{c}}$) and determine the latest start time for this set (i.e., $LS_{\mathbf{b},\mathbf{c}}$) based on the minimum processing capacity

7b. Create the set of blocks that represents the union of the holding sets for blocks $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ (i.e., $H_{\mathbf{a},\mathbf{b},\mathbf{c}}$) and determine the latest start time for this set (i.e., $\tilde{\tau} = LS_{\mathbf{a},\mathbf{b},\mathbf{c}}$) based on the minimum processing capacity

8b. Determine the latest that any two-block set must be accessed:

$$\tilde{\tau}' = \max\left(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}}\right)$$

9b. Determine the latest that any single block must be accessed:

$$\tilde{\tau}'' = \max\left(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}}\right)$$

10b. If $\tilde{\tau} < \tilde{\tau}'$ then create a cut of the form:

$$w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1$$

11b. If $\tilde{\tau}' < \tilde{\tau}''$ then create a cut of the form:

$$w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2$$

Output all generated cuts

Relative Dominance of Three-Way Latest Starts Cuts As with two-way cuts, the time index ($\tilde{\tau}$ or $\tilde{\tau}'$) in our generated cuts tells us about the relative dominance of different cuts. Cuts with an earlier time index dominate cuts (involving the same blocks) with a later time index.

Three-way cuts concern themselves with another dominance issue though; the value to the right of the inequality. Take the following two potential cuts:

- $w_{\mathbf{a},2} + w_{\mathbf{b},2} + w_{\mathbf{c},2} \geq 1$
- $w_{\mathbf{a},2} + w_{\mathbf{b},2} + w_{\mathbf{c},2} \geq 2$

The first means that at least one of the three blocks **a**, **b**, and **c** must be mined by time period 2, while the second means that at least two of these three blocks must be mined by time period 2. In this case, the latter cut dominates the former. The reason for this is that the latter is more restrictive than the former. The latter requires that two blocks be mined, while the former requires only one of the blocks be mined. Our three-way earliest starts cuts algorithm accounts for this dominance and only generates the dominant cut for any set of blocks in the same time period (assuming such a cut is valid and useful).

Three-Way Latest Starts Cuts Numerical Example Looking at our two-dimensional example, we use blocks 9, 11, and 13 to create a three-way latest starts cut based on the minimum production capacity (see Figure 4.7).

For this example, we assume that:

- Block **a** is represented by block 9 in the figure, block **b** is represented by block 11 in the figure, and block **c** is represented by block 13 in the figure
- Each block contains 10 tons of material (i.e., $n_b = 10$ for each block **a**, **b**, and **c**)
- The minimum production requirement is 20 tons per time period (for simplicity, we only use production bounds for this example)

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block (blocks 9, 11, and 13 in the figure above for this example) and the t index identifies the time period by which the block is extracted. We now use the algorithm to generate a three-way latest starts cut based on the minimum production requirement:

1. Determine the latest start time for block **a**:

   $LS_{\mathbf{a}} = 9$

2. Determine the latest start time for block **b:**

   $LS_{\mathbf{b}} = 9$

3. Determine the latest start time for block **c:**

   $LS_{\mathbf{c}} = 9$

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.7. Three-Way Latest Starts Cuts Numerical Example. This example depicts the results of creating a three-way latest starts cut with blocks 9, 11, and 13.

4. Create the set of blocks that represents the union of the holding sets for blocks **a** and **b** and determine this set's latest start time:

   $H_{\mathbf{a},\mathbf{b}} = H_{\mathbf{a}} \cup H_{\mathbf{b}} = \{9, 15, 16, 17\} \cup \{11, 17, 18, 19\} = \{9, 11, 15, 16, 17, 18, 19\}$

   $LS_{\mathbf{a},\mathbf{b}} = 8$

5. Create the set of blocks that represents the union of the holding sets for blocks **a** and **c** and determine this set's latest start time:

   $H_{\mathbf{a},\mathbf{c}} = H_{\mathbf{a}} \cup H_{\mathbf{c}} = \{9, 15, 16, 17\} \cup \{13, 19, 20, 21\} = \{9, 13, 15, 16, 17, 19, 20, 21\}$

   $LS_{\mathbf{a},\mathbf{c}} = 7$

6. Create the set of blocks that represents the union of the holding sets for blocks **b** and **c** and determine this set's latest start time:

   $H_{\mathbf{b},\mathbf{c}} = H_{\mathbf{b}} \cup H_{\mathbf{c}} = \{11, 17, 18, 19\} \cup \{13, 19, 20, 21\} = \{11, 13, 17, 18, 19, 20, 21\}$

   $LS_{\mathbf{b},\mathbf{c}} = 8$

7. Create the set of blocks that represents the union of the holding sets for blocks **a**, **b**, and **c** and determine this set's latest start time:

   $H_{\mathbf{a},\mathbf{b},\mathbf{c}} = H_{\mathbf{a}} \cup H_{\mathbf{b}} \cup H_{\mathbf{c}} = \{9, 15, 16, 17\} \cup \{11, 17, 18, 19\} \cup \{13, 19, 20, 21\} = \{9, 11, 13, 15, 16, 17, 18, 19, 20, 21\}$

   $\tilde{\tau} = LS_{\mathbf{a},\mathbf{b},\mathbf{c}} = 6$

8. Determine the latest that any two-block set can be accessed:

   $\tilde{\tau}' = \max(LS_{\mathbf{a},\mathbf{b}}, LS_{\mathbf{a},\mathbf{c}}, LS_{\mathbf{b},\mathbf{c}}) = \max(8,7,8) = 8$

9. Determine the latest that any single block can be accessed:

   $\tilde{\tau}'' = \max(LS_{\mathbf{a}}, LS_{\mathbf{b}}, LS_{\mathbf{c}}) = \max(9,9,9) = 9$

10. Since $\tilde{\tau} < \tilde{\tau}'$ we can create a cut of the form:

    $w_{\mathbf{a},\tilde{\tau}} + w_{\mathbf{b},\tilde{\tau}} + w_{\mathbf{c},\tilde{\tau}} \geq 1 \qquad \Rightarrow \qquad w_{9,6} + w_{11,6} + w_{13,6} \geq 1$

11. Since $\tilde{\tau}' < \tilde{\tau}''$ we can create a cut of the form:

$$w_{\mathbf{a},\tilde{\tau}'} + w_{\mathbf{b},\tilde{\tau}'} + w_{\mathbf{c},\tilde{\tau}'} \geq 2 \qquad \Rightarrow \qquad w_{9,8} + w_{11,8} + w_{13,8} \geq 2$$

This means that by the end of time period 6, at least one of the three blocks represented by blocks 9, 11, and 13 in the figure above must be mined. Also, by the end of time period 8, at least two of the three blocks represented by blocks 9, 11, and 13 in the figure above must be mined.

4.3.6 Cuts Involving More than Three Blocks

The general ideas we present in our two-block and three-block cut generating algorithms can be extended to create cuts of more than three blocks. In fact, we can generate cuts involving as many blocks as can be both produced and processed in any given time period. However, creating cuts with more than three blocks gets harder and more time consuming. As the number of blocks involved in the cut increases, the number of possible block combinations that must be investigated grows exponentially. This makes it harder to find these cuts yet still requires that they be worth the time investment with respect to the reduction in overall problem solve time. Our empirical evidence suggests that cuts become less effective as the number of blocks they contain increases.

4.3.7 Using the *by* vs *at* Formulation in Cut Generation

We employ the *by* formulation to create our cuts. It is worth mentioning that if we use the alternative *at* formulation, then the cuts we describe above are not sufficient to produce the desired effects. For instance, say a valid and useful cut for the *by* formulation is:

$$w_{3,3} + w_{4,3} \leq 1 \tag{4.1}$$

Using the *by* formulation of the problem, the variable w_{bt} represents block b being mined *by* time period t. Therefore this one constraint implies that *by* time period

3, *at most* one of the blocks 3 or 4 can be mined. The *by* formulation implicitly accounts for all time periods up to and including the time period represented by the t subscript, so the constraint in equation (4.1) above implicitly addresses what occurs in time periods 1 and 2, along with time period 3 (which is the t subscript). However, if we use the *at* formulation, the constraint in equation (4.1) above only requires that blocks 3 and 4 cannot both be mined *at* time period 3. For time periods later than time period 3, this constraint suffices, but for time periods 1 and 2, there is a problem with the formulation.

In order to get the same result with the *at* formulation, we need to pursue one of two approaches. Either we require a set of three constraints or a cut involving summation. Recall that the decision variable in the *at* formulation is y_{bt}, representing block b being mined *at* time period t.

A set of three constraints of the form:

$$\begin{aligned} y_{3,1} + y_{4,1} &\leq 1 \\ y_{3,2} + y_{4,2} &\leq 1 \\ y_{3,3} + y_{4,3} &\leq 1 \end{aligned} \tag{4.2}$$

accomplishes the same thing as the constraint in equation (4.1). These constraints limit the removal of blocks 3 and 4 to at most one *at* time periods 1, 2, and 3. Since the *at* formulation includes constraints that permit blocks to be mined no more than once during the time horizon via constraints (3.1) and (3.2), these three constraints generate the same cut as equation (4.1). As the t index gets closer to the end of the time horizon (i.e., approaches T), the number of constraints required to create a cut using the *at* formulation increases. With the *by* formulation, however, we only need one constraint for each cut, regardless of the t index on the constraint.

Another approach involves using summation notation. A constraint of the form:

$$\sum_{u=1}^{3} y_{3,u} + \sum_{u=1}^{3} y_{4,u} \leq 1$$

suffices to define the constraint represented in equation (4.1). Here again, as the t index gets close to the end of the time horizon, the number of terms being summed increases accordingly. In general, using the *at* formulation adds more complexity to the generation of valid and useful cuts, thus further justifying our use of the *by* formulation.

4.4 Lagrangian Relaxation Methods

Lagrangian relaxation methods attempt to *move* complicating constraints to the objective function, thus leaving a set of constraints that are *relatively easily* adhered to. The relaxed constraints are dualized and added to the objective function with fixed penalties (i.e., Lagrange multipliers usually denoted by λ's with various subscripts as indices).

In the block sequencing problem, the *side* constraints that enforce minimum and maximum operational bounds tend to complicate the otherwise *simple* structure of the problem and are therefore considered complicating constraints. These *side* constraints include:

- Average grade requirements
- Mine production capacity constraints
- Mill processing capacity constraints

With respect to the *by* formulation presented in Section 3.3.2, these *side* constraints correspond to constraints (3.11), (3.12), (3.13), and (3.14), respectively (note that the average grade constraints are written as two separate constraints for formatting

reasons). Because we assume a fixed cutoff grade in our model, we do not include the average grade constraints (constraints (3.11) and (3.12)) in our formulation. We move the production and processing constraints to the objective function and add weights (i.e., Lagrange multipliers) to discourage violations.

We introduce Lagrange multipliers indexed by constraint type, i.e., minimum processing, minimum production, maximum processing, and maximum production (corresponding to the i index) and time period (corresponding to the t index). We call our Lagrangian multipliers λ_{it} ($i = 1 \ldots 4$, $t = 1 \ldots T$). The resultant Lagrangian relaxation formulation is:

$$\begin{aligned}
\max \sum_{b \in B} \sum_{t \in T} c_{bt} \left(w_{bt} - w_{b,t-1} \right) \quad &+ \quad \sum_{t \in T} \lambda_{1t} \left(\sum_{b \in B} n_b \left(w_{bt} - w_{b,t-1} \right) - \underline{E} \right) \\
&+ \quad \sum_{t \in T} \lambda_{2t} \left(\overline{E} - \sum_{b \in B} n_b \left(w_{bt} - w_{b,t-1} \right) \right) \\
&+ \quad \sum_{t \in T} \lambda_{3t} \left(\sum_{b \in B} r_b \left(w_{bt} - w_{b,t-1} \right) - \underline{C} \right) \\
&+ \quad \sum_{t \in T} \lambda_{4t} \left(\overline{C} - \sum_{b \in B} r_b \left(w_{bt} - w_{b,t-1} \right) \right) \qquad (4.3)
\end{aligned}$$

$$\begin{aligned}
\text{subject to:} \quad & w_{b,t-1} \leq w_{bt} \, \forall \, b, t > 1 && (4.4) \\
& \sum_{t \in T} \left(w_{bt} - w_{b,t-1} \right) = 1 \, \forall \, b \ni \arg\max T_b \leq |T| && (4.5) \\
& \sum_{t \in T} \left(w_{bt} - w_{b,t-1} \right) \leq 1 \, \forall \, b \ni \arg\max T_b > |T| && (4.6) \\
& w_{bt} \leq w_{b't} \, \forall \, b \in B, b' \in B_b, t && (4.7) \\
& w_{bt} \in \{0, 1\} \ \forall \, b, t && (4.8)
\end{aligned}$$

Notice that the *side* constraints (3.13) and (3.14) are now in the objective function,

(4.3), each prefixed by its own Lagrange multiplier, λ_{it}. This leaves only five sets of constraints in the problem, thus significantly simplifying the resulting problem's structure.

Not all of the *side* constraints must be moved to the objective function; we can selectively choose which ones to move. Because there are four side constraints involved (a minimum and maximum production constraint and a minimum and maximum processing constraint), there are 16 combinations of scenarios that we investigate with respect to dualizing these constraints. The simplest scenarios involve moving only one of these constraints (since there are four side constraints, there are four such scenarios). There are six scenarios which move two of these constraints, four scenarios that move three of these constraints, and lastly one scenario that moves all four of these constraints to the objective function. The scenario that moves none of these constraints to the objective function is our monolith.

4.4.1 Basic Idea Behind the Lagrangian Relaxation Method

Implementing the Lagrangian relaxation procedure in AMPL and solving it with CPLEX involves the use of a script to control execution of the program between the monolith and the Lagrangian relaxation subproblem. Our iterative process attempts to tighten lower and upper bounds on the optimal objective function value by successively solving the Lagrangian relaxation subproblem of the monolith and using that solution (if it is feasible) in the monolith to determine an optimal extraction schedule for that iteration.

Lagrangian relaxation starts by solving the linear programming (LP) relaxation of the monolith and using that objective function value as an initial upper bound on the monolith objective function value. Subsequent iterations solve the Lagrangian relaxation subproblem of the monolith. The optimal decision variable values from the Lagrangian relaxation subproblem are simply inserted into the monolith (assuming they are feasible in the monolith) to derive a new monolith objective function value.

If the objective function value for the current iteration's Lagrangian relaxation subproblem, z^*_{LR}, is less than the incumbent upper bound for the monolith, then we update the upper bound with z^*_{LR}. If the current iteration's monolith objective function value, z^*_{mono}, is higher than the incumbent lower bound for the monolith, then we update the lower bound with z^*_{mono}. Before the next iteration, the Lagrangian relaxation procedure updates the Lagrangian multipliers based on the degree of violation incurred by each of the dualized constraints in the monolith. We terminate the procedure after either reaching an iteration limit or achieving a small enough gap between the Lagrangian procedure's lower and upper bounds.

As mentioned above, to obtain the initial upper bound, we solve the LP-relaxation of the monolith. In our case, this upper bound is actually quite tight, especially as the data sets get bigger. Since our problem closely resembles a constrained knapsack, we look at some characteristics of this class of problems to understand this phenomenon. With bigger data sets, we have the ability to use more heterogeneous left-hand-side coefficients to fill our knapsack capacity constraints. Two very simple constrained knapsack problems help illustrate this principle. First, examine the following problem:

$$\max \ 10x_1 + 10x_2 + 10x_3 \tag{4.9}$$

$$\text{subject to} : 10x_1 + 10x_2 + 10x_3 \leq 25 \tag{4.10}$$

$$x_i \in \{0, 1\} \tag{4.11}$$

The optimal LP-relaxation solution is $x_1^{LP} = 1$, $x_2^{LP} = 1$, $x_3^{LP} = \frac{1}{2}$, resulting in an optimal objective function value of $z^*_{LP} = 25$. The optimal integer programming (IP) solution is $x_1^{IP} = 1$, $x_2^{IP} = 1$, $x_3^{IP} = 0$, resulting in an optimal objective function value of $z^*_{IP} = 20$. For this problem, the LP-relaxation does not provide a very strong

upper bound. We now present a second problem:

$$\max \ 3x_1 + 5x_2 + 4x_3 + 7x_4 + 8x_5 \tag{4.12}$$

$$\text{subject to} : 3x_1 + 5x_2 + 4x_3 + 7x_4 + 8x_5 \leq 25 \tag{4.13}$$

$$x_i \in \{0, 1\} \tag{4.14}$$

The optimal LP-relaxation solution for this problem is $x_1^{LP} = \frac{1}{3}$, $x_2^{LP} = 1$, $x_3^{LP} = 1$, $x_4^{LP} = 1$, $x_5^{LP} = 1$, resulting in an optimal objective function value of $z_{LP}^* = 25$. The optimal IP solution is $x_1^{IP} = 0$, $x_2^{IP} = 1$, $x_3^{IP} = 1$, $x_4^{IP} = 1$, $x_5^{IP} = 1$, resulting in an optimal objective function value of $z_{IP}^* = 24$. For this second problem, the LP-relaxation provides a very strong upper bound. The second problem typifies our larger data sets; they contain more data that is more heterogeneously distributed with respect to total material and valuable ore content. As such, for our larger data sets, the initial LP-relaxation to the monolith provides a very good upper bound.

The key to success when using the Lagrangian relaxation method is selecting the correct constraints(s) to dualize and then properly setting the values of the multipliers for these dualized constraints (the λ_{it}'s, in our case). Selecting inappropriate constraints to dualize may not lead to a simplified Lagrangian relaxation subproblem or might result in Lagrangian relaxation subproblem solutions that are never feasible in the monolith. If the dualized multipliers are too high, then the constraints with which they are associated may have too much slack and the solution may be suboptimal. On the other hand, if these multipliers are too low, then their associated constraints may be violated (since the cost of violation isn't high enough) and the solution may be infeasible for the monolith. There are various methods employed in the literature to update the Lagrangian multipliers between successive iterations, some of which are discussed in Section 4.4.2.

Among the most troublesome aspects of Lagrangian relaxation is the problem of infeasibility. Generally speaking, the Lagrangian relaxation subproblem has no diffi-

culty finding a solution; however, that solution may not be feasible in the monolith. Because some constraints from the monolith are dualized in the Lagrangian relaxation subproblem, the solution to the Lagrangian relaxation subproblem may allow for constraint violations, especially if the cost of doing so (based on the Lagrangian multiplier values) is low. Despite attempts at modifying the Lagrangian multiplier values to discourage constraint violations, there may be no way to obtain a feasible solution to the monolith problem, resulting in what is known as the "condition of gaps" (Dagdelen 1985, pp. 99-100). The mathematical explanation for the existence of these gaps has to do with the fact that the mapping of solutions between the Lagrangian relaxation subproblem and the monolith may not be *onto* (Everett 1963):

> The Lagrange multiplier method therefore generates a mapping of the space of lambda vectors (components λ^k, $k = 1, ..., n$) into the space of constraint vectors (components c^k, k = 1, ..., n [where c^k represent the constraints in the monolith]). There is no *a priori* guarantee, however, that this mapping is onto—for a given problem there may be inaccessible regions (called *gaps*) consisting of constraint vectors that are not generated by any λ vectors. (p. 407)

As Fisher points out (1985, p. 18) "In my experience, it is rare in practice that the Lagrangian solution will be feasible in the original problem. However, it is not uncommon that the Lagrangian solution will be nearly feasible and can be made feasible with some minor modifications." Our experience concurs with this statement and we find that we rarely obtain a Lagrangian relaxation subproblem solution that is feasible for the monolith, especially as the number of constraints we dualize increases. However, following Fisher's notion that these infeasible solutions can be made feasible, we create a *feasing* routine that endeavors to do exactly this.

Dagdelen (1985) solves the block scheduling problem by Lagrangian decomposition techniques. He employs subgradient methods to modify the Lagrangian multipli-

ers corresponding to the side constraints consisting of blending and capacity requirements in his problem formulation. He further reduces the resulting multi-time period Lagrangian relaxation subproblem into a series of efficiently-solvable single time period problems. He exploits the network structure of the sequencing constraints in these single time period problems and uses adjusted block values to solve each of these problems as an ultimate pit limits problem via the Lerchs-Grossman algorithm, ultimately creating a block extraction schedule for the entire ore body.. Dagdelen overcomes the "condition of gaps" by allowing the operational side constraints which he dualizes in the Lagrangian relaxation subproblem to be violated by "ϵ" in the primal (i.e., the monolith). Doing this means that the constraints in the formulation of his monolith are elastic, an idea we do not employ. Without this allowable ϵ error, his procedure would continue to attempt to create a nonexistent feasible solution and would result in an endless loop. As a result, the constraint would never be met and a feasible solution to the monolith would not be found.

Kawahata (2006) expands on the Lagrangian relaxation procedure developed by Dagdelen (1985). His methodology uses two Lagrangian relaxation subproblems, one to represent the most aggressive production scheduling case (i.e., working at the maximum production bound) and the other to represent the most conservative production sequencing case (i.e., working at the minimum production bound), to restrict the monolith's optimal solution space. The premise is that the decision variable values from the solutions to these two Lagrangian relaxation subproblems eliminate variables from the monolith, thus significantly speeding up solve times. However, he still contends that "a gap problem cannot be avoided as long as the Lagrangian relaxation method is applied to solve the production scheduling problem." (2006, p. 64) We show that there are cases in which the optimal decision variable values to the Lagrangian relaxation subproblem are feasible in the monolith. When the optimal decision variable values to the Lagrangian relaxation subproblem are not feasible in the monolith, we attempt to make them feasible via our *feasing* routine, thus eliminating

the gap problem.

4.4.2 Implementation of the Lagrangian Relaxation Method for our Problem

Unlike Dagdelen (1985) and Kawahata (2006), we treat our constraints as having rigid right-hand-sides; thus, we do not allow constraint violations. Although more realistic, this means we are plagued by an infeasible monolith solution once the Lagrangian relaxation subproblem generates a potential solution. As such, we endeavor not only to intelligently update our multiplier values, but also to employ heuristics to force our solutions to be feasible. The success of our *feasing* routine depends on the characteristics of the data, but ultimately its use aids in resolving the infeasibility issues with which the Lagrangian relaxation procedure is beset.

Steps in the Lagrangian Relaxation Technique The Lagrangian relaxation technique we employ consists of an iterative process which attempts to place lower and upper bounds on our monolith's optimal objective function value. The steps in the procedure are:

1. Solve the LP relaxation of the monolith – this serves as the initial upper bound (UB).

2. Solve the Lagrangian relaxation subproblem (LRSP).

   If the LRSP's optimal objective function value is less than UB, then update UB.

3. Insert the LRSP's optimal decision variable values into the monolith (assuming they are feasible for the monolith).

   If the monolith's optimal objective function value is greater than LB, then update LB.

4. Update the Lagrangian multipliers (the λ_{it}'s).

5. Return to step 2 unless:

    - iteration limit is reached
    - acceptable LB-UB gap is reached

Successfully implementing the Lagrangian relaxation procedure is dependent on many issues. First and foremost, we need good initial values for the Lagrangian multipliers. Then we must have an efficient and effective means to update these Lagrangian multipliers between iterations. Lastly, we must have a method for generating feasible solutions for our monolith (if the optimal decision variable values from the Lagrangian relaxation subproblem are not feasible in the monolith). We address all of these issues and propose ways of resolving them to ensure that the Lagrangian relaxation procedure converges to an acceptable solution quickly.

Scenarios Our model formulation consists of four sets of side constraints: 1) minimum production, 2) minimum processing, 3) maximum production, and 4) maximum processing. As mentioned in Section 4.4, these four constraints result in 15 scenarios representing the various one-way, two-way, three-way, and four-way combinations to dualize them for use in the Lagrangian relaxation procedure. Generally speaking, the more constraints we dualize, the simpler the structure of the resulting Lagrangian relaxation subproblem becomes. Dualizing just one constraint simplifies the monolith's constraint set only slightly, because there are still three other complicating side constraints with which the solver must contend. Dualizing all four constraints means that the resulting Lagrangian relaxation subproblem has a simplified structure that solves quickly.

Unfortunately, as more constraints are dualized in the Lagrangian relaxation subproblem, more constraints are violated in the monolith. If only one constraint is dualized, then the resulting solution from the Lagrangian relaxation subproblem

still has three other constraints that help bound the solution and make it feasible (or near feasible) for the monolith. When all four constraints are dualized, there is nothing besides the sequencing constraints (constraints (4.4)) and the four other auxiliary constraints (constraints (4.5), (4.6), (4.7), and (4.8)) to force the Lagrangian relaxation subproblem's solution to create a feasible solution for the monolith. As a result, the Lagrangian relaxation subproblem's solution is generally highly infeasible in the monolith in the sense that many constraints are violated and the extent of these violations is large.

Using our *feasing* routine we can force a feasible solution if the infeasibility is not too great (i.e., the number of infeasible constraints is low and/or the extent to which they are violated is not too great). Discouragingly, if the infeasibility is great, even with our *feasing* routine, we cannot generate a feasible solution for the monolith. The latter is what frequently happens if we dualize three or four of the constraints. Additionally, the actual amount of time spent conducting the *feasing* routine becomes excessive, thus negating all the time savings achieved in solving the much simplified Lagrangian relaxation subproblem. Empirically, we see the best results in terms of quickly converging to an acceptable answer by dualizing only one or two constraints and then using our *feasing* routine on the Lagrangian relaxation subproblem's solution to create a feasible solution for the monolith.

Multiplier Maximum Values Generally, the only constraints imposed on the multipliers used in the Lagrangian relaxation method are that they be non-negative. Essentially, the higher the multipliers' values, the more penalty there is to the objective function value for violating the multiplier's associated constraint. However, there is a high enough multiplier value above which the associated dualized constraint is not violated because the penalty to the objective function value is greater than the penalty incurred by violating the associated dualized constraint. Raising the constraint's multiplier value beyond this maximum results in over-penalizing the ob-

jective function value, leading to a lower objective function and a poorer lower bound. Excessively punishing the objective function with multiplier values that are too high leads to wasted iterations, since the multipliers need to be adjusted downward over the course of subsequent iterations. Therefore, we use information from the problem formulation to set maximum multiplier values.

When dualizing the maximum production and/or processing constraints, the multipliers' upper bounds equal the maximum profit per ton that any accessible block could possibly achieve. This value is the *maximum* profit because that is the most we would be willing to pay to violate the constraint. Any value higher than this is excessive punishment for violating the constraint. We calculate this value by determining the maximum ratio between the discounted block value and the total tonnage of the block $\left(\max_{\{b,t\}}\left(\frac{c_{bt}}{n_b}\right)\right)$, being sure only to include those blocks that are actually accessible by time period t (based on the block's earliest start time).

On the other hand, when dualizing the minimum production and/or processing constraints, the multipliers' upper bounds equal the most profit per ton foregone for any accessible block. Again, this value is the most we would be willing to pay to violate the constraint, but in this case it represents how much we would be willing to pay to not have to mine undesirable blocks. We calculate this value by determining the minimum ratio between the discounted block value and the total tonnage of the block $\left(\min_{\{b,t\}}\left(\frac{c_{bt}}{n_b}\right)\right)$, again being sure only to include those blocks that are accessible by time period t (based on the block's earliest start time). If all the blocks being investigated are ore blocks, then this value is some positive number and the minimum processing and production constraints are never violated. As such, the maximum value for the multipliers should be 0 since we don't need to punish the objective for a constraint that is never violated.

Multiplier Seeding The ultimate goal of the Lagrangian relaxation procedure is to derive an optimal solution to the Lagrangian relaxation subproblem that is also

feasible in the monolith and approaches an optimal solution for the monolith. To do this, we must discern optimal multiplier values. Through an iterative process, we adjust the multiplier values based on their effect on their respective dualized constraints in the monolith (i.e., the amount of slack that the dualized constraints contain as a result of the solution from the Lagrangian relaxation subproblem). Since with each iteration we are refining an educated guess, the initial value we use to seed the multipliers can have a dramatic effect on the number of iterations we must conduct in order to generate a solution within an acceptable margin or error.

Although zero may be used to seed the multiplier values, we find that this value is not very helpful. Essentially, seeding the multipliers with zero means that there is no punishment in the objective function for violating the associated constraints. Since our objective function is based on a net present value analysis, such a scheme results in many ore blocks being mined at their earliest possible times, thus severely violating the maximum production and processing constraints. Because of these constraint violations, the multiplier values must be increased, and during the early iterations of the Lagrangian relaxation procedure, the multipliers are constantly alternating to correct under utilizing and violating the dualized constraint, creating a structure that is not feasible for the monolith. Our experience shows that an initial value of 1 works much better for our problem formulation. Seeding the multipliers with a value of 1 means that we incur some degree of punishment for violating a constraint, but that punishment is not overly severe.

Another scheme is to seed the multipliers with the dual values from their associated constraints in the LP-relaxation. We can generate these dual values when we solve the LP-relaxation of the monolith to create our initial upper bound for the problem. However, for large data sets, the time spent finding these dual values is not trivial. Additionally, if the constraints are not tight in the LP relaxation of the monolith, the resultant duals are zero, which means that the dualized constraints are not punished at all (see discussion above). Overall, our experience does not indicate

that seeding multipliers with their duals is very useful.

Multiplier Updating Routines As mentioned in Section 4.4.1, there are various methods available to update the multiplier values. Among the most common is one called the *subgradient method*. As Fisher (1981, p. 7) points out, "The subgradient method is a brazen adaptation of the gradient method in which gradients are replaced by subgradients." Given an initial value for the multiplier, λ_{it}^0, generate a sequence of multipliers using the rule:

$$\lambda_{it}^{k+1} = \lambda_{it}^k + t^k(Ax^k - b) \quad \forall\ i, t, k$$

where t^k is a positive scalar representing the step size at iteration k and the term $(Ax^k - b)$ represents the slack in the dualized constraint at iteration k as a result of finding the optimal solution to the Lagrangian relaxation subproblem at iteration k. In the same paper, Fisher points out two other popular approaches for updating Lagrangian multipliers: 1) those employing the simplex method via column generation techniques and 2) multiplier adjustment methods. The former do not see much use because they tend to converge slowly and are rather difficult to program. The latter, which are problem-specific, may afford great benefits if used properly.

To solve our problem via the Lagrangian relaxation method, we use the subgradient method (which we call the *traditional* approach to multiplier updating) and also attempt some multiplier adjustment methods employing either a percentage change (i.e., we increase or decrease the multiplier by a fixed percentage at each iteration) or the traditional approach with a historical look-back (i.e., we set the new multiplier value equal to the weighted average of its value in the previous iteration as well as what it *traditionally* would be in the current iteration).

The percentage change multiplier adjustment method works as follows: If an inequality constraint is satisfied, then the multiplier value encourages the objective

function to utilize slack in this constraint. On the other hand, if the constraint is not met, then the value of the multiplier for that constraint is raised by a fixed percentage as a way of discouraging the Lagrangian relaxation subproblem's objective function from violating that constraint.

The traditional approach with a historical look-back initially uses the subgradient method described above to create a new multiplier value. However, the resultant multiplier value is not used in its entirety. A fixed percentage of the previous iteration's multiplier value is included along with a fixed percentage of the current iteration's multiplier value (calculated via the subgradient method) to create the updated multiplier value for the current iteration:

$$\lambda_{it}^{k} \Leftarrow (\text{curr}\%)\, \lambda_{it}^{k} + (\text{hist}\%)\, \lambda_{it}^{k-1} \quad \forall\, i, t, k$$

For example, 75% of the current iteration's multiplier and 25% of the previous iteration's multiplier may be used to create the current iteration's multiplier value. As a result, this method employs information from the previous iteration to help prevent large changes in multiplier values, especially in the first few iterations of the Lagrangian relaxation method.

Held, Wolfe, and Crowder (1974) present another multiplier updating scheme based on four different scaling parameters; however, we do not find their methods very promising.

4.4.3 Feasing Routines

Fisher's idea of modifying an infeasible solution so that it becomes feasible is what leads us to propose a *feasing* routine for the open pit scheduling problem. Given enough *spare* blocks not in the current Lagrangian relaxation subproblem's optimal solution (i.e., blocks that are not mined in the current optimal solution), we can selectively add or remove blocks from the optimal Lagrangian relaxation subproblem's

solution to create a solution that is feasible for the monolith. When infeasibilities occur because of not meeting minimum production and/or processing bounds, we simply *add* the *best* blocks to the solution to ensure that these lower bounds are met. On the other hand, when infeasibilities occur because of exceeding the maximum production and/or processing bounds, we *remove* the *best* blocks from the solution to ensure that the upper bounds are met. In both cases, by *best* we mean that we pick blocks that help us meet the various constraint bounds using as few blocks as possible and avoid violating other constraint bounds in the process. We start the *feasing* process in the first time period with a constraint violation and then check all subsequent time periods to ensure that our *feasing* actions in previous time periods do not have adverse affects. If our *feasing* actions from previous time periods cause subsequent time period's constraints to become infeasible, we use our *feasing* routine on these later time periods also. Once our *feasing* routine is complete, we ensure that the solution to the Lagrangian relaxation subproblem is feasible for all time periods before passing the decision variable values back to the monolith. Our empirical experience shows that employing this *feasing* routine significantly increases our ability to use the Lagrangian relaxation method and determine feasible solutions for the monolith (see Section 5.3.2 for results).

When using the *feasing* routine to eliminate infeasibilities due to violating the maximum processing or maximum production, by *best*, we mean removing those blocks that best help meet the maximum constraints while not violating the minimum constraints or the sequencing constraints in the process. Our goal is to generate a feasible solution for the monolith. For example, if our current Lagrangian relaxation subproblem solution violates maximum production constraints, then the *feasing* routine finds the heaviest waste block(s) to remove. Removing *waste* blocks ensures that we do not violate the maximum processing constraints (because waste blocks have no processable material in them). Picking the *heaviest* waste blocks means that we do not spend extra time searching for more blocks than necessary in order to

meet the maximum production constraint. To ensure that we do not violate any of the sequencing constraints, we remove blocks from among those that represent the bottom-most *mined* blocks in the current time period's optimal solution to the Lagrangian relaxation subproblem. By *remove* we mean that we mine the block one time period later, unless we are at the end of the time horizon, in which case the block is not mined at all. Selecting a correct block in the current time period whose extraction we shift to one time period later ensures that we do not adversely affect the sequencing constraints for subsequent time periods. Also, we are careful not to isolate a block on any level, thus violating the sixth sequencing constraint. If our actions result in leaving a block completely alone on a given level, we also move that block to the next time period so that the we do not violate the sixth sequencing constraint. The example in Section 4.4.3 clarifies this concept.

When we conduct the *feasing* routine to remove infeasibilities as a result of minimum processing or minimum production constraint violations, by *best* we mean adding blocks that best help meet the minimum constraints without violating the maximum constraints in the process. For example, if the current Lagrangian relaxation subproblem solution violates the minimum processing constraints, then the *feasing* routine finds the heaviest ore block(s) to add. Adding *ore* blocks ensures that we make the violated minimum processing constraint feasible while not adding useless waste blocks that may potentially create a violation of the maximum production constraint. Again, picking the *heaviest* ore blocks ensures that we do not search for more blocks than necessary to satisfy the minimum processing constraint. The blocks are added at the top of the pit, picking among those blocks that are not in the optimal solution by the current time period (i.e., we *mine* unmined blocks that are at the highest level in the pit). Adding blocks at the top of the pit guarantees that we do not violate the sequencing constraints and preserves the feasibility of the solution with respect to the sequencing constraints when used in the monolith. Again, we ensure that no blocks are isolated on any level so that we do not violate the sixth

sequencing constraint. If our added block is isolated on a given level, then we also add a neighbor block to preclude violating the sixth sequencing constraint. We terminate the feasing routine when we obtain a feasible solution, or when there are no more blocks to shift. An example in Section 4.4.3 clarifies this concept.

Although our feasing routine helps produce feasible solutions, there are some caveats. First and foremost, there are some Lagrangian relaxation subproblem solutions that contain constraint violations to such a degree that our feasing routine cannot correct them. This is especially true as the number of time periods increases and/or the number of dualized constraints increases. The feasing routine may also take a long time to execute, especially for large data sets.

Feasing Routine for Maximum Constraints Algorithm

Assumptions We include all the assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- T = number of time periods in the horizon
- t = time period in which a maximum constraint is violated
- $B_t^{eligible}$ = the set of all blocks that are on the lowest level (with respect to the z-axis) of the Lagrangian relaxation subproblem's optimal solution in time period t
- k_t^{low} = the z-coordinate of the set $B_t^{eligible}$ in time period t
- w_{bt}^{best} = the variable representing the *best* block b (i.e., b^{best}) to remove from the Lagrangian relaxation subproblem's optimal solution in time period t for the current iteration

Inputs

- Maximum processing capacity per time period (in tons of ore) and maximum production capacity per time period (in tons of material) – note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized)
- An optimal solution for the Lagrangian relaxation subproblem, $\left\{w_{bt}^{LR}\right\}$
- A set of blocks not in the optimal solution for the Lagrangian relaxation subproblem, $\left\{\overline{w}_{bt}^{LR}\right\}$

Outputs

- A feasible solution for the monolith based on the current optimal solution from the Lagrangian relaxation subproblem

Algorithm

For all time periods $t = 1...T$ repeat while the solution is infeasible in the monolith or until further *feasing* routine actions cannot be taken, i.e., until $B_t^{eligible}$ is empty:

1. Determine $B_t^{eligible}$ and k_t^{low} for time period t.
2. Determine the best block to remove, w_{bt}^{best}:

   If any of the following three scenarios occurs:

   - the maximum production constraint is violated
   - the maximum production constraint is violated by a greater percentage than the maximum processing constraint
   - the maximum production constraint and the minimum processing constraint are both violated

   Then:

let w_{bt}^{best} = the heaviest block (with respect to weight) of all the blocks in $B_t^{eligible}$ that contain the least amount of usable material in them

If any of the following three scenarios occurs:

- the maximum processing constraint is violated
- the maximum processing constraint is violated by a greater percentage than the maximum production constraint
- the maximum processing constraint and the minimum production constraint are both violated

Then:

let w_{bt}^{best} = the heaviest ore block (with respect to usable material) of all the blocks in $B_t^{eligible}$ that contain the least amount of total material in them

3. Set $w_{bt}^{best} = 0$ (i.e., *not mined* based on the definition of our variables)
4. If $t < T$ then mine block b^{best} in the next time period (i.e., set $w_{b,t+1}^{best} = 1$)
5. Ensure that the sixth sequencing constraint is not violated by moving the block w_{bt}^{best} to the next time period.

   If the block w_{bt}^{best} is alone on level k_t^{low} in time period t (i.e., it has no neighbors) then it can be moved without violating the sixth sequencing constraint. Otherwise, check if each one of its plus sign neighbors has a neighbor. If moving w_{bt}^{best} to the next time period isolates any one of its neighbors, then the isolated neighbor block must also be moved to the next time period.

6. Check the current solution to ensure it is feasible for the monolith in time period t, i.e., it satisfies the operational (side) constraints in time period t. If the solution is feasible, increment t by 1 and return to step 1. Otherwise, go to step 7.

7. Update the set $B_t^{eligible}$ by removing b^{best} from it, and correspondingly update k_t^{low}, if applicable. Return to step 2.

If the algorithm produces a feasible solution for the monolith, use this solution in the monolith to attempt to update its lower bound.

Feasing Routine for Maximum Constraints Numerical Example Using our two-dimensional example, we employ our feasing routine for maximum constraints to create a feasible solution for the monolith from an infeasible solution generated by the Lagrangian relaxation subproblem (see Figure 4.8 below).

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.8. Feasing Routine for Maximum Constraints Numerical Example. This example depicts the idea of using the feasing routine to render an infeasible Lagrangian relaxation subproblem solution feasible for the monolith by removing the *best* block among blocks 18 and 19.

For this example, we assume that:

- Each block contains 10 tons of material (i.e., $n_b = 10$)
- The maximum production constraint is 40 tons per time period
- $t = 3$ and $t < T$
- The z-coordinate index runs from 1 at the bottom to 3 at the top of the pit

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block and the t index identifies a time period by which the block is

extracted. With a maximum production capacity of 40 tons per time period, by the end of time period 3 at most 120 tons of material can be mined. However, the diagram above depicts 130 tons of material being mined by the end of time period 3 (all the light grey and dark grey blocks), so the solution to the Lagrangian relaxation subproblem violates the maximum production constraint in the monolith. We use the feasing routine to find the *best* block that is part of the optimal solution to the Lagrangian relaxation subproblem in time period 3 and remove it from the solution:

1. Determine $B_t^{eligible}$ and k_t^{low} for time period 3:

   $B_3^{eligible} = \{18, 19\}$ and therefore $k_3^{low} = 1$.

2. Determine w_{bt}^{best}:

   Since the maximum production constraint is the only violated constraint, $w_{b,3}^{best}$ represents the heaviest block (with respect to weight) of the the blocks in $B_3^{eligible}$. For this example, however, each block weighs the same (10 tons), so we arbitrarily choose block 18 as the *best* block: $w_{b,3}^{best} \Rightarrow w_{18,3}^{best}$.

3. Set $w_{bt}^{best} = 0$:

   $w_{18,3}^{best} = 0$

4. If $t < T$ then mine block b^{best} in the next time period:

   Since we assume $t < T$, then $w_{18,4}^{best} = 1$.

5. Ensure that the sixth sequencing constraint is not violated by moving w_{bt}^{best} to the next time period:

   Moving block 18 to be mined in time period 4 now isolates block 19, so in order to obey the sixth sequencing constraint, we must move block 19 to be mined in time period 4 also.

6. Now we have a feasible solution in time period 3. Let $t = 4$ and return to step 1 (in the case that the solution is infeasible in time period 4 or later).

After eliminating infeasibilities for time periods 4 through T, we can use the modified optimal solution to the Lagrangian relaxation subproblem in the monolith to generate an objective function value (an NPV) and update the lower bound if the resultant value is greater than the incumbent lower bound.

Feasing Routine for Minimum Constraints Algorithm

Assumptions We again include all the assumptions that we describe with respect to our model formulation (see Section 3.1).

Definitions

- T = number of time periods in the horizon
- t = time period in which a minimum constraint is violated
- $B_t^{eligible}$ = the set of all blocks that are on the highest level (with respect to the z-axis) of the Lagrangian relaxation subproblem's optimal solution in time period t
- k_t^{high} = the z-coordinate of the set $B_t^{eligible}$ in time period t
- w_{bt}^{best} = the variable representing the *best* block b (i.e., b^{best}) to add to the Lagrangian relaxation subproblem's optimal solution in time period t for the current iteration

Inputs

- Minimum processing capacity per time period (in tons of ore) and minimum production capacity per time period (in tons of material) – note that these capacity constraints must be hard constraints (i.e., they cannot be elasticized)
- An optimal solution for the Lagrangian relaxation subproblem, $\left\{w_{bt}^{LR}\right\}$

- A set of blocks not in the optimal solution for the Lagrangian relaxation subproblem, $\left\{\overline{w}_{bt}^{LR}\right\}$

Outputs

- A feasible solution for the monolith based on the current optimal solution from the Lagrangian relaxation subproblem

Algorithm

For all time periods $t = 1...T$ repeat while the solution is infeasible in the monolith and there are blocks that can be added to the solution in time period t (i.e., there are blocks in the data set *not mined* that can be mined in time period t based on their earliest start times):

1. Determine $B_t^{eligible}$ and k_t^{high} for time period t.
2. Determine the best block to add, w_{bt}^{best}:

   If any of the following three scenarios occurs:

   - the minimum production constraint is violated
   - the minimum production constraint is violated by a greater percentage than the minimum processing constraint
   - the minimum production constraint and the maximum processing constraint are both violated

   Then:

   let w_{bt}^{best} = the heaviest block (with respect to weight) of all the blocks in $B_t^{eligible}$ that contain the least amount of usable material in them

   If any of the following three scenarios occurs:

   - the minimum processing constraint is violated

- the minimum processing constraint is violated by a greater percentage than the minimum production constraint
- the minimum processing constraint and the maximum production constraint are both violated

Then:

let w_{bt}^{best} = the heaviest ore block (with respect to usable material) of all the blocks in $B_t^{eligible}$ that contain the least amount of total material in them

3. Set $w_{bt}^{best} = 1 \ \forall t = t+1, \ldots, T$ (i.e., *mined* based on the definition of our variables)
4. Ensure that the sixth sequencing constraint is not violated by adding the block w_{bt}^{best} to the optimal solution for time period t.

   If adding the block w_{bt}^{best} to the optimal solution for time period t means that the block w_{bt}^{best} is alone (i.e., has no plus-sign neighbors) on level k_t^{high}, then its addition to the optimal solution violates the sixth sequencing constraint. To avoid violating the sixth sequencing constraint when adding w_{bt}^{best} to the optimal solution, also add the best block from among $w_{bt}^{best\prime}$s plus-sign neighbors to the optimal solution.
5. Check the current solution to ensure it is feasible for the monolith in time period t, i.e., it satisfies the operational (side) constraints in time period t. If the solution is feasible, increment t by 1 and return to step 1. Otherwise, go to step 6.
6. Update the set $B_t^{eligible}$ by removing b^{best} from it, and correspondingly update k_t^{low}, if applicable. Return to step 2.

If the algorithm produces a feasible solution for the monolith, use this solution in the monolith to attempt to update its lower bound.

Feasing Routine for Minimum Constraints Numerical Example Using our two-dimensional example, we employ our feasing routine for minimum constraints to create a feasible solution for the monolith from an infeasible solution generated by the Lagrangian relaxation subproblem (see Figure 4.9 below).

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Figure 4.9. Feasing Routine for Minimum Constraints Numerical Example. This example depicts the idea of using the feasing routine to render an infeasible Lagrangian relaxation subproblem solution feasible for the monolith by adding the *best* block among blocks 8, 9, and 14.

For this example, we assume that:

- Each block contains 10 tons of material (i.e., $n_b = 10$)
- All blocks have an earliest start time of $t = 1$
- The minimum production constraint is 20 tons per time period
- $t = 7$
- The z-coordinate index runs from 1 at the bottom to 3 at the top of the pit

Recall that our decision variables are defined as w_{bt}, where the b index identifies the particular block and the t index identifies a time period by which the block is extracted. With a minimum production capacity of 20 tons per time period, by the end of time period 7 at least 140 tons of material must be mined. However, the diagram above depicts 130 tons of material being mined by the end of time period

7 (the light grey blocks), so the solution to the Lagrangian relaxation subproblem violates the minimum production constraint in the monolith. We use the feasing routine to find the *best* block that is not part of the optimal solution to the Lagrangian relaxation subproblem in time period 7 and add it to the solution:

1. Determine $B_t^{eligible}$ and k_t^{high} for time period 7:

   $B_7^{eligible} = \{8, 9, 14\}$ and therefore $k_7^{high} = 2$.

2. Determine w_{bt}^{best}:

   Since the minimum production constraint is the only violated constraint, $w_{b,7}^{best}$ represents the heaviest block (with respect to weight) of the the blocks in $B_7^{eligible}$. For this example, however, each block weighs the same (10 tons), so we arbitrarily chose block 8 as the *best* block: $w_{b,7}^{best} \Rightarrow w_{8,7}^{best}$.

3. Set $w_{bt}^{best} = 1$:

   $w_{8,7}^{best} = 1$

4. Ensure that the sixth sequencing constraint is not violated by adding w_{bt}^{best} to the optimal solution for time period t:

   Adding block 8 to the optimal solution in time period 7 means adding an isolated block, thus violating the sixth sequencing constraint. As a result, we must also add block 9 to the optimal solution for time period 7 in order to obey the sixth sequencing constraint.

5. Now we have a feasible solution in time period 7. Let $t = 8$ and return to step 1 (in the case that the solution is infeasible in time period 8 or later).

After eliminating infeasibilities for time periods 8 through T, we can use the modified optimal solution to the Lagrangian relaxation subproblem in the monolith to generate an objective function value (an NPV) and update the lower bound if the resultant value is greater than the incumbent lower bound.

Chapter 5

NUMERICAL RESULTS

5.1 Data

To examine the methods and procedures described, we use a master data set that represents an open pit mine consisting of 19,320 blocks to empirically test our methodologies. Associated with this data set are minimum and maximum bounds on the per time period production and processing constraints at the mine. The mine follows 45° sloping rules.

We reduce this 19,320 block data set by an order of magnitude into an envelope of blocks that contains 1,060 blocks. We create yet another data set two orders of magnitude smaller called a micro-pit. This micro-pit includes 196 blocks. With such a small data set, we are able to obtain an optimal solution and graphically investigate the results.

To further investigate our methodologies, we create two additional data sets from the master data set. The first of these is a data set containing the 1,980 blocks found in a 13 by 13 by 12 block subset of the original 19,320 blocks. The second of these data sets is one containing the 2,880 blocks in an 18 by 17 by 12 block subset of the original 19,320 blocks. Note that the original 19,320 block data set is not a uniform cube of blocks, so the subsets we create are also not uniform cubes (this is why $13 \times 13 \times 12 \neq 1,980$ and $18 \times 17 \times 12 \neq 2,880$). When describing our computational results, we refer to these various data sets by the number of blocks they contain.

To obtain additional large data sets, we perturb the mineral content of each of the blocks in the 19,320 block data set by $\pm 5\%$ to create seven additional instances of this large data set. We refer to these data sets as *A*, *B*, *C*, *D*, *E*, *F*, and *G* perturbations.

Lastly, we examine a much more complicated open pit mining model with variable cut-off grades, stockpiles, and blocks that can be partially mined. We refer to this data set as *Newmont* and use it to show how our methodologies work in a more general setting.

Below is a table of all the various data sets we use and their pertinent characteristics, including the number of time periods in the horizon:

name	# blocks	# binary variables	# constraints	# time periods
1,060	1,060	6,360	32,748	6
1,980	1,980	11,880	71,562	6
2,880	2,880	17,280	105,504	6
10,819	10,819	64,914	395,885	6
10,819A	10,819	64,914	395,885	6
10,819B	10,819	64,914	395,885	6
10,819C	10,819	64,914	395,885	6
10,819D	10,819	64,914	395,885	6
10,819E	10,819	64,914	395,885	6
10,819F	10,819	64,914	395,885	6
10,819G	10,819	64,914	395,885	6
Newmont	61	1,391	55,022	25

Table 5.1. Data Sets Used to Empirically Test our Methodologies. This table summarizes the pertinent characteristics of the various data sets we employ to test our solution methodologies

It is important to note that the *Newmont* formulation also includes 162,934 continuous variables.

5.1.1 Data Pre-processing

We find that examining the data we use in our model formulations before actually running the optimization routines provides some very enlightening insights. Some of the data we use has many individual datum that we can effectively remove from the data set without sacrificing optimality in any manner. In practice, using data blindly without investigating its characteristics can either lead to erroneous results or extra computation.

Our largest data set containing 19,320 blocks is 25 blocks wide in the x-coordinate direction by 26 blocks long in the y-coordinate direction by 60 blocks deep in the z-coordinate direction (note that the z-coordinate runs from the bottom up, i.e., a lower number represents a deeper level in the pit). Although a complete block structure with these dimensions would have 39,000 blocks in it (25 * 26 * 60 = 39,000), this data set has only half that many blocks indicating that some pre-processing has been done to eliminate blocks that are not part of the orebody.

Examining the resulting 19,320 blocks even further yields the observation that there are absolutely no ore blocks on any of the bottom 17 levels of the pit and therefore no reason to include any of these 5,474 blocks in our data set since they will never be mined. Because of this, we reduce our 19,320 block data set to a 13,846 block data set. Next, we discover that dispersed throughout the rest of the ore body are 3,027 *phantom* blocks which are completely empty (they contain no material and no mineral content). Removing these blocks leads us to a data set containing 10,819 blocks, thus nearly halving the size of our original 19,320 block data set. This reduction in the data set pays huge dividends for all our solution methodologies, especially since integer programming is notoriously plagued by exponential solve times with respect to the size of problem instance.

5.2 CPLEX Parameter Settings

We use the AMPL programming language, version 2006.06.26 (2006) to formulate our model. We then enter this formulation into the CPLEX solver, version 10.1 (2006). CPLEX offers many parameter settings that can be altered by the user when solving mixed integer programming problems. Varying these parameter settings can dramatically change the problem's solution time. Unfortunately, there are many parameters to explore and no one combination of settings works for all problems. The model's performance depends on the combination of parameter settings used in the CPLEX solver. As such, we must either determine the best parameter settings

for each problem instance or find a set of parameters that works well for different instances of the same class of problems. The most straightforward way to determine which parameter settings to use is to try each setting available. However, this quickly becomes an enormous task. To efficiently discern which parameter settings work best (including how the various settings interact with each other), we rely on past programming experience and knowledge of what seems to work well with similarly formulated problems to investigate those parameters that have generated the most promising results. Based on the problem being investigated here, the most promising parameters (and their associated definitions according to the AMPL CPLEX 10.0 User's Guide by ILOG 2006) are:

- baropt - used to specify the barrier (i.e., interior point) method to solve linear programming problems
- branch - used to specify a branching direction on the fractional decision variable value (i.e., strong or weak branching)
- heurfreq - frequency with which CPLEX applies a rounding heuristic at the nodes
- mipcuts - used to specify the level of aggression CPLEX uses to generate cuts based on different combinatorial constructs
- mipemphasis - used to guide CPLEX's branch and cut strategy
- probe - used to determine the amount of solution probing CPLEX conducts
- rinsheur - used to determine how often to apply the relaxation induced neighborhood search heuristic (RINS heuristic)

These seven parameters and their various settings result in 216 different combinations we explore to discern the best parameter settings. The results are different for each

data set used. However some general trends do emerge. The following parameter settings produce the fastest solution times for the class or problems we investigate:

- monolith without earliest starts, latest starts, or cuts — branch -1 mipcuts -1 mipemphasis 1 probe 1 rinsheur 40
- monolith with earliest starts and latest starts — branch -1 mipcuts -1 mipemphasis 1 probe 1 rinsheur 40
- monolith with earliest starts, latest starts, and cuts — mipcuts -1 mipemphasis 1 rinsheur 40
- Lagrangian relaxation procedure — baropt[1] branch -1 heurfreq 20 mipemphasis 1 rinsheur 40

The newest version of CPLEX, version 11, has a parameter tuning feature which intelligently selects the best parameter settings to use for each problem instance. Future work on this problem would benefit from its use.

5.3 Computational Results

We use a Sun Fire V240 with 2GB of RAM to conduct all computations in CPLEX. We use an IBM Thinkpad with an Intel 2.13 GHz processor and 2.0 GB of RAM and a LENOVO desktop with dual core AMD ATHLON64 5000+ processors and 2.0 GB of RAM to run all earliest starts, latest starts, and cuts algorithms.

5.3.1 Visual Depiction of an Extraction Sequence

As discussed in Section 5.1, we create a micro version of the data to investigate different aspects of the model. Since the micro-pit is so small, graphing the results of a two time period problem instance is relatively easy and representing the actual

[1]Note that the baropt parameter only pertains to the initial LP relaxation. We do not use it to solve any of the Lagrangian relaxation subproblems.

three-dimensional pit outlines during each time period is possible (see Figure 5.1 below). This dynamic depiction of how the actual extraction operations occur at the mine helps mine engineers communicate the schedule to their employees.

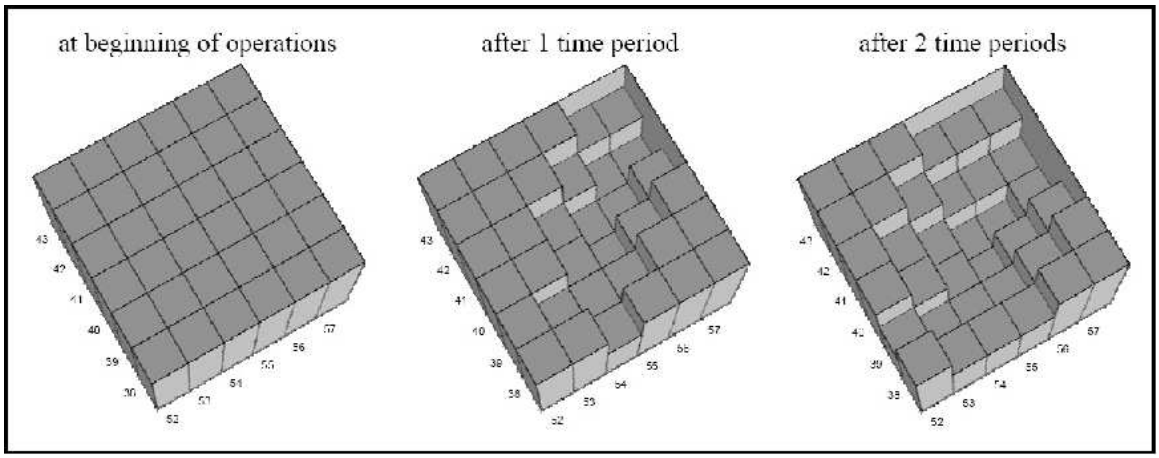

Figure 5.1. Visual Depiction of Micro Pit Results. This figure depicts the optimal solution of a two-period extraction sequence using the micro-pit data.

5.3.2 Computational Results for Earliest Starts, Latest Starts, Cuts, and the Lagrangian Relaxation Procedure

We use our earliest starts procedure to calculate a complete predecessor list for each block in the data set and then determine each block's earliest start time period. Next, we use our latest starts procedure to calculate a complete holder list for each block in the data set and then determine each block's latest start time period. Lastly, we use these predecessor and holder lists, along with their associated earliest and latest starts to generate cuts. When generating cuts, we use our *reasonable block selection rule* to empirically determine how many blocks to include for the various types of cuts we generate. Generating cuts involves a degree of judgment and must be balanced with the amount of time it takes to solve the monolith. Generally speaking, we aim to generate cuts to such a degree that their generation time is no more than about 20% of the time it takes to solve the monolith without any earliest or latest

starts or cuts. We summarize these generation times in Table 5.2 below:

problem instance	preds time (sec.)	ES time (sec.)	holders time (sec.)	LS time (sec.)	cuts time (sec.)
1,060	10	3	11	3	62
1,980	14	4	15	4	46
2,880	22	8	24	8	284
10,819	1,059	61	N/A	N/A	1,869
10,819A	1,130	65	N/A	N/A	1,816
10,819B	1,090	63	N/A	N/A	1,834
10,819C	1,128	65	N/A	N/A	1,803
10,819D	1,120	64	N/A	N/A	1,802
10,819E	1,125	65	N/A	N/A	1,798
10,819F	1,305	75	N/A	N/A	2,471
10,819G	1,285	74	N/A	N/A	2,404
10,819 AVG	1,155	67	N/A	N/A	1,975
Newmont	~ 0	~ 0	N/A	N/A	4

Table 5.2. Summary of Generation Times for Predecessor Lists, Earliest Starts, Holder Lists, Latest Starts, and Cuts. This table summarizes the time spent to generate predecessor lists (*preds*) and the associated earliest starts (*ES*), holder lists (*holders*) and the associated latest starts (*LS*), and cuts (*cuts*). All times are in seconds. The penultimate row in the table (*10,819 AVG*) presents the average results for all the 10,819 block data set instances. We do not create holder lists nor latest starts for the 10,819 data set instances and the Newmont data set.

Looking at Table 5.2, we notice that for our largest data set, on average we spend one minute to calculate the earliest starts for all 10,819 blocks. We do not create holder lists nor latest starts for any of the 10,819 data set instances because empirical evidence shows us that these values are not useful due to the characteristics of these data sets.

When applying the Lagrangian relaxation procedure, it is important to dualize the correct constraint(s). Our empirical evidence indicates that dualizing more than one constraint results in a Lagrangian relaxation subproblem whose optimal decision variable values are not feasible in the monolith. Additionally, we are not able to make the optimal solution to the Lagrangian relaxation subproblem feasible with our *feasing* routine, rendering the entire Lagrangian relaxation procedure in its current

implementation ineffective. As such, we conclude that dualizing only one constraint works best. The actual constraint to dualize is dependent on the nature of the data. Our results indicate that dualizing the constraints with the most slack in them works best. However, it is easy to dualize each constraint and then run the Lagrangian procedure in parallel on four separate machines. The first instance to converge to an acceptable solution gap indicates which single constraint to dualize.

We use our *feasing* routine in any iteration that involves a Lagrangian relaxation subproblem solution that is infeasible in the monolith. Additionally, we conduct our *feasing* routine until we either find a feasible solution to the monolith or run out of blocks with which we can conduct the feasing routine (i.e., if there are no more blocks to add to or remove from the model, the *feasing* routine terminates).

Using the *by* formulation we describe in Section 3.3.2 above, we calculate a solution within 2% of optimality to determine the extraction sequence for the data sets presented in Table 5.1 above. We present detailed results in the appendix and a summary of the execution times in Table 5.3 below.

Examining Table 5.3, it is apparent that our methodologies drastically improve solution times. We compare the monolith's solution time (column *monolith* in Table 5.3) with the solution times of using just earliest and latest starts (column *ES & LS* in Table 5.3), earliest and latest starts with cuts (column *ES & LS & cuts* in Table 5.3), and the Lagrangian relaxation procedure with earliest and latest starts (column *Lagrangian Relaxation with ES & LS* in Table 5.3). Overall, using earliest and latest starts reduces computer solve times by 80.2%. Including cuts with earliest and latest starts also results in an average solution time reduction of 80.2%. Implementing the Lagrangian relaxation procedure provides an average reduction of 84.0%.

Taking the size of the data sets into account, we see that the Lagrangian relaxation procedure significantly improves solution times for bigger data sets, while earliest and latest starts with cuts seem to work best with smaller data sets. First and foremost, none of the eight instances of the 10819 data set solve to 2% mipgap

problem instance	monolith solution time (sec.)	ES & LS solution time (sec.)	ES & LS & cuts solution time (sec.)	Lagrangian Relaxation with ES & LS solution time (sec.)
1,060	1,082	147	82	105
1,980	202	148	95	193
2,880	1,481	366	419	660
10,819	$>$ 24 hrs.	16,326	3,801	708
10,819A	$>$ 24 hrs.	1,570	7,803	769
10,819B	$>$ 24 hrs.	4,225	29,545	1,261
10,819C	$>$ 24 hrs.	3,054	1,570	698
10,819D	$>$ 24 hrs.	4,355	1,633	1,075
10,819E	$>$ 24 hrs.	1,426	2,794	680
10,819F	$>$ 24 hrs.	3,680	7,273	14,159
10,819G	$>$ 24 hrs.	838	14,121	3,687
10,819 AVG	$>$ 24 hrs.	4,434	8,567	2,880
Newmont	11,476	9,719	8,696	N/A

Table 5.3. Summary of Results from Implementing Earliest Starts, Latest Starts, Cuts, and the Lagrangian Relaxation Procedure. This table compares the results (in seconds of CPLEX solve time) of using our algorithms on the various data set instances. The column labeled *monolith* represents the raw data. The column labeled *ES & LS* is the raw data with earliest and latest starts implemented. The column labeled *ES & LS & cuts* depicts the raw data with earliest and latest starts and an appropriate level of cuts included. Lastly, the column labeled *Lagrangian Relaxation with ES & LS* presents the results from implementing the Lagrangian relaxation procedure on the data set with earliest and latest starts. The penultimate row in the table (*10,819 AVG*) presents the average results for all the 10,819 block data set instances. The structure of the Newmont formulation is different enough from our model formulation to preclude us from using the Lagrangian relaxation procedure on it.

within 24 hours (i.e., 86400 seconds). For these large data sets, we observe that earliest and latest starts reduce solve times by 94.9%, while adding cuts actually decreases this time savings to about 90.1%. However, the Lagrangian relaxation procedure works extremely well, reducing solve times by an average of 96.7%. For six of the eight 10819 data instances, the Lagrangian relaxation procedure is the fastest method, reducing solve times by an astonishing 99%.

5.3.3 Comparison of Results with Commercial Software

To gain an appreciation for how well our methodologies work, we compare our results with those of a commercially available mine scheduling software package, MineSight Economic Planner – MSEP (2006). MSEP uses the traditional approach to solve the block sequencing problem, so it first determines the ultimate pit limits and then generates nested pits and pushbacks which it uses to schedule the block extraction of the ore body. The software suffers from many limitations, including 1) an inability to specify a time horizon, 2) not being able to include lower bounds on the operational constraints, and 3) a failure to adhere to upper bounds on the operational constraints. The software claims to use a dynamic cutoff grade approach, which theoretically provides a better schedule with respect to maximizing NPV.

We use the 1060, 1980, 2880, and 10819 (original case only) to test MSEP's performance. We present the results from using MSEP for these four data instances in Table 5.4 below: In each instance, the software violates the upper bound on the production constraint in time period 1. Looking at Table 5.4, these violations are not trivial, amounting to over a ten-fold increase in required production capacity for the 10,819 data set instance. Additionally, depending on the data set, the software arbitrarily chooses a time horizon: 4 time periods for the 1060 data set, 7 time periods for the 1980 data set, 7 time periods for the 2880 data set, and 10 time periods for the 10819 data set. Despite using a dynamic cutoff grade, disregarding maximum production constraints, not adhering to minimum operational constraints,

and arbitrarily setting a longer time horizon in all but the 1060 block data set, MSEP's optimal NPVs are lower for the 1060, 1980, and 2880 block data sets, and only marginally higher for the 10819 data set. Recall, though, that the 10819 block data set is run for 10 time periods by MSEP, while ours is only 6 time periods.

The actual algorithm that MSEP uses is a complete mystery. Granted, the software runs remarkably quickly (solution times are on the order of 20 seconds or fewer), but we have absolutely no confidence in the quality of these solutions. The heuristic MSEP employs has serious drawbacks that result in unimplementable extraction schedules. Specifically, the ten-fold violation of the maximum production constraint in time period 1 assumes that a certain amount of waste can be removed, i.e., *pre-stripped*, during a "pre-production" year. This requires that resources are available for such an activity. Additionally, there is no indication that the software includes the cost of conducting this initial work in its NPV calculation.

We are confident in the quality of our solutions. We use deterministic operations research methods that have withstood the test of time. Our solution times may be longer, but the resultant block extraction schedule adheres to all operational and

problem instance	our NPV ($\$10^6$)	MSEP NPV ($\$10^6$)	# time periods	time period 1 production constraint violation (tons)
1,060	19.0	13.5	4	773,000
1,980	17.5	17.3	7	600,000
2,880	15.5	14.2	7	3,000,000
10,819	9.1	9.8	10	10,500,000

Table 5.4. MSEP's Results for the 1,060, 1,980, 2,880, and 10,819 Data Set Instances. This table summarizes the results of implementing the 1,060, 1,980, 2,880, and 10,819 data set instances in MSEP. The column labeled *Our NPV* shows the NPV we achieve using the same data set in our monolith MIP formulation. The column labeled *MSEP NPV* presents MSEP's final NPV for the extracted blocks from the pit. The column labeled *# time periods* shows how many time periods MSEP uses to calculate its NPV. The last column shows how many excess tons of production capacity (over the 1,000,000 tons stipulated by the maximum production capacity constraint) MSEP requires to achieve the NPV it reports.

geospatial constraints and we have a bound on the solution's quality. Waiting longer for a feasible solution is worthwhile.

Chapter 6

LIMITATIONS, EXTENSIONS, AND CONCLUSION

6.1 Limitations and Extensions

Our model is first and foremost a deterministic model. All inputs are known with absolute certainty. In real-world mines, this assumption is often not valid, especially with regard to block characteristics deep underground. To effectively address the random nature of block content or the value of the ore being removed from the mine, stochastic programming is a better approach for solving realistic open pit mining problems. As Dimitrakopoulos (1998) shows, the optimal solution to the block sequencing problem is affected by uncertainties in many of the input parameters such as: 1) in-situ grade uncertainty, 2) uncertainty in the operational mining specifications such as production and processing capacities or sloping rules, and 3) economic uncertainties with respect to operating costs or the value of the mineral being extracted. In order to address these potential uncertainties, Achireko and Frimpong (1996) use neural networks to resolve the randomness in block characteristics while Ramazan and Dimitrakopoulos (2004b) directly address in-situ grade variability in their model formulation.

Variable cutoff grade models should also be investigated. Such models add another dimension to the decision variables (in the form of a location index (l) indicating where each block is sent in the optimal solution), but more accurately reflect reality and handle *in-situ* ore variability better. However, adding another index significantly increases the number of decision variables that must be investigated by the model. The extra problem detail this affords comes at a cost of larger problem size.

There may be ways to generate earliest and latest start times based on the

minimum or maximum number of blocks that can be removed in a certain time period. Tighter problem formulations may be achieved using more aggressive cut generation schemes that investigate larger sets of blocks. A stronger reasonable block selection rule or quicker cut generation methods would allow us to include more cuts in the formulation without significantly increasing overall solution time (i.e., the time spent generating the cuts would not be absorbed by the time saved in using them).

With respect to the Lagrangian relaxation procedure, there are many additional tactics that may be employed. Nemhauser and Wolsey (1988) provide some alternatives to the subgradient method for multiplier updating. Among these alternative methods is one that uses a constraint generation idea which could also lead to the introduction of more cuts via cutting planes, thus further tightening the formulation. Additionally, the use of the interior point method for the solution of the LP relaxation of the monolith or for any other linear programs may provide solutions with a different and promising algebraic structure.

The *feasing* routine we create may also be improved. Determining how long to conduct the feasing routine and how often (with respect to iteration count) it should be used are items warranting further investigation. The *feasing* routine might only be applied every n iterations, or only if the Lagrangian relaxation solution is not "too infeasible" based on both the number of constraints violated in the monolith and the extent to which these violations occur. Lastly, placing a limit on the amount of time spent conducting the *feasing* routine may preclude using it for excessive amounts of time with little or no success.

A completely different *feasing* routine based on something other than manually adding or removing blocks from the Lagrangian relaxation subproblem solution may also be worth examining. One such *feasing* routine may be to find the first time period with a constraint violation and impose the violated constraint as a hard constraint, then resolve the Lagrangian relaxation subproblem and check if these new decision variable values are now feasible in the monolith, imposing the violated con-

straint as another hard constraint if the solution is not feasible in the monolith. Such an incremental approach, however, may take a long time to implement because we must resolve the Lagrangian relaxation subproblem after each constraint is added. Ultimately, any routine that endeavors to render feasible those Lagrangian relaxation subproblems that are infeasible in the monolith must allow the Lagrangian relaxation procedure to converge to within an acceptable margin of error faster than solving the monolith outright.

Aggregating time periods or blocks to determine strategic mine schedules may also be useful. Other relaxation and decomposition techniques, such Dantzig-Wolfe decomposition and column generation methods, may provide fruitful results.

Additional research to reduce solution times should focus on methods to either limit the number of variables in the problem or methods that do not necessitate the use of branch-and-bound algorithms to solve MIP problem instances. Heuristics based on genetic algorithms or artificial neural networks may provide better solution times.

Regardless of the methodology used, any procedure that can reduce the solution time required to determine the efficient block extraction schedule is of benefit to mine engineers in their quest to efficiently sequence the extraction of profitable material from their mines. The alternative is to either suffer with slow algorithms and long solution times, or use intuition to guess the best extraction schedule. Neither alternative is attractive for the complex mines that we see in the world today.

6.2 Conclusion

Efficiently scheduling the extraction of ore from an open pit mine helps ensure that the mine maximizes the net present value of the minerals in the orebody. Solving the block sequencing problem results in a time-indexed schedule of when any given block in the orebody should be removed (if it is removed at all) that maximizes the NPV of the ore in the pit subject to all sequencing and operational constraints.

Mine planners use two approaches to solving the block sequencing problem: one

based on the ultimate pit limits and another based on a comprehensive approach. The former divides the process into three separate stages that are solved sequentially, while the later takes a global view of the problem. Although more difficult to solve, the comprehensive approach provides more flexibility and ultimately creates a better schedule. In our research, we pursue solving the block sequencing problem using this approach. We propose various methodologies that make the problem more tractable. We limit the solution space by defining decision variables only between their earliest and latest possible start times. We present a series of cut generation algorithms that produce valid and useful cuts to tighten the problem formulation. Lastly, we employ a Lagrangian relaxation technique with a *feasing* routine to make infeasible Lagrangian relaxation subproblem solutions feasible for the monolith.

Employing our methodologies significantly reduces solve times while not compromising the optimal solution. Although our earliest starts idea appears in the literature, no one employs a latest starts idea in open pit mining. Our cuts are much more aggressive. The Lagrangian relaxation procedure we use does not require soft constraints, but instead uses our *feasing* routine to ensure feasible Lagrangian relaxation subproblem solutions for the monolith.

The techniques we present: 1) earliest and latest starts, 2) cuts, and 3) Lagrangian relaxation, serve as tools to expedite solution times for the block sequencing problem. Just like any handyman knows, one tool is never sufficient for all jobs. In the same vein, our three tools complement each other and serve as different techniques to aid in arriving at solutions to the block sequencing problem. Our empirical results show that using our tools reduces solve times by well over 95% without sacrificing any confidence in the answers achieved. In today's finicky commodities markets, being able to adapt to changing market conditions and incorporate the latest mine-specific data to update operating schedules is paramount to ensure a profitable mining venture.

REFERENCES

Achireko, P. and Frimpong, S., 1996. "Open pit optimization using artificial neural networks on conditionally simulated blocks." In R. Ramani, editor, *Proceedings of the 26th Symposium on the Application of Computers and Operations Research in the Mineral Industry (26th APCOM)*, Society for Mining, Metallurgy, and Exploration, Inc, Littleton, CO, chapter 44. pp. 285–290.

Akaike, A. and Dagdelen, K., 1999. "A strategic production scheduling method for an open pit mine." In K. Dagdelen, editor, *28th APCOM*, Society for Mining, Metallurgy, and Exploration, Inc, Littleton, CO. pp. 729–738.

Alford, C., 1995. "Optimization in underground mine design." In *25th APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO. pp. 213–218.

AMPL, 2006. "AMPL version 10.1 2006.06.26."

Bertsimas, D. and Stock-Patterson, S., 1998. "The air traffic flow management problem with enroute capacities." *Operations Research*, Vol. 46, No. 3, pp. 406–422.

Bertsimas, D. and Tsitsiklis, J., 1997. "Ch 12.4 - the air traffic flow management problem." In *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts, chapter 12.4. pp. 544–551.

Boland, N., Dumitrescu, I., Froyland, G., and Gleixner, A., 2007. "LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity." *Computers and Operations Research*, Vol. doi:10.1016/j.cor.2007.12.006.

Boland, N., Fricke, C., and Froyland, G., 2006. "A strengthened formulation for the open pit mine production scheduling problem." Department of Mathematics and Statistics, University of Melbourne, Victoria, Australia, 3010.

Caccetta, L. and Hill, S., 2003. "An application of branch and cut to open pit mine scheduling." *Journal of Global Optimization*, Vol. 27, pp. 349–365.

Cai, W., 1989. *Application of Network Flow and Zero-One Programming to Open-Pit Mine Design Problems*. Ph.D. thesis, University of Arizona, Tucson, AZ.

Cai, W., 2001. "Design of open-pit phases with consideration of schedule constraints." In H. Xie, Y. Wang, and Y. Jiang, editors, *29th APCOM*, Swets & Zeitlinger B.V., Lisse, the Netherlands. pp. 217–221.

CPLEX, 2006. "CPLEX version 10.1."

Dagdelen, K., 1985. *Optimum Multi Period Open Pit Mine Production Scheduling*. Ph.D. thesis, Colorado School of Mines.

Dagdelen, K., 2005. "Open pit optimization - strategies for improving economics of mine planning." In R. Dimitrakopoulos, editor, *Orebody Modeling and Strategic Mine Planning - Uncertainty and Risk Management Models*, Australasian Institute of Mining Metallurgy, Carlton Victoria, Australia, Vol. 14. pp. 125–128.

Denby, B. and Schofield, D., 1994. "Open-pit design and scheduling by use of genetic algorithms." *Transactions of the Institution of Mining and Metallurgy (Section A: Mining Industry)*, Vol. 103, pp. A21–A26.

Dimitrakopoulos, R., 1998. "Conditional simulation algorithms for modelling orebody uncertainty in open pit optimization." *International Journal of Mining, Reclamation and Environment*, Vol. 12, No. 4, pp. 173–179.

Dowd, P. and Onur, A., 1992. "Optimizing open pit design and sequencing." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 42. pp. 411–422.

Elevli, B., 1995. "Open pit mine design and extraction sequencing by use of OR and AI concepts." *International Journal of Surface Mining Reclamation and Environment*, Vol. 9, pp. 149–153.

Erarslan, K. and Çelebi, N., 2001. "A simulative model for optimum open pit design." *CIM Bulletin*, Vol. 94, pp. 59–68.

Espinoza, D., Goycoolea, M., Moreno, E., and Rubio, E., 2008. "Topological sorting heuristics for the resource-constrained open-pit mining problem." Working Paper, University of Chile.

Everett, H., 1963. "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources." *Operations Research*, Vol. 11, pp. 399–417.

Fisher, M., 1981. "The Lagrangian relaxation method for solving integer programming problems." *Management Science*, Vol. 27, No. 1, pp. 1–18.

Fisher, M., 1985. "An applications oriented guide to Lagrangian relaxation." *Interfaces*, Vol. 15, No. 2, pp. 10–21.

Froyland, G., Menabde, M., Stone, P., and Hodson, D., 2004. "The value of additional drilling to open pit mining projects." In *Proceedings of Orebody Modeling and Strategic Mine Planning - Uncertainty and Risk Management*. Perth, Australia, pp. 225–232.

Gershon, M., 1982. "A linear programming approach to mine scheduling optimization." In T. Johnson and R. Barnes, editors, *17th APCOM*, Society of Mining Engineers of the American Institute of Mining, Metallurgical, and Petroleum Engineers, Inc., New York, NY, chapter 43. pp. 483–493.

Gershon, M., 1987. "Heuristic approaches for mine planning and production scheduling." *International Journal of Mining and Geological Engineering*, Vol. 5, pp. 1–13.

Halatchev, R., 2002. "The time aspect of the optimum long-term open pit production sequencing." In S. Bandopadhyay, editor, *30th APCOM*, Society of Mining, Metallurgy, and Exploration, Inc, Littleton, CO. pp. 133–146.

Held, M., Wolfe, P., and Crowder, H., 1974. "Validation of subgradient optimization." *Mathematical Programming*, Vol. 6, pp. 62–88.

Hochbaum, D., 2001. "A new-old algorithm for minimum-cut and maximum-flow in closure graphs." *Networks*, Vol. 37, No. 4, pp. 171–193.

Hochbaum, D. and Chen, A., 2000. "Performance analysis and best implementations of old and new algorithms for the open-pit mining problem." *Operations Research*, Vol. 48, No. 6, pp. 894–914.

Hoerger, S., Bachmann, J., Criss, K., and Shortridge, E., 1999. "Long term mine and process scheduling at Newmont's Nevada operations." In K. Dagdelen, editor, *28th APCOM*, Society for Mining, Metallurgy, and Exploration, Inc, Littleton, CO. pp. 739–748.

Hoffman, R. and Ball, M., 2000. "A comparison of formulations for the single-airport ground-holding problem with banking constraints." *Operations Research*, Vol. 48, No. 4, pp. 578–590.

Hulse, D., 1992. "The consequences of block size decisions in ore body modeling." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 22. pp. 225–232.

Huttagosol, P. and Cameron, R., 1992. "A computer design of ultimate pit limit by using transportation algorithm." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 45. pp. 443–460.

ILOG, 2006. *ILOG AMPL CPLEX System Version 10.0 User's Guide*. ILOG, Mountain View, CA.

Johnson, T., 1968. *Optimum Open Pit Mine Production Scheduling*. Ph.D. thesis, University of California, Berkeley.

Johnson, T., 1969. "Optimum open-pit mine production scheduling." In A. Weiss, editor, *A Decade of Digital Computing in the Mineral Industry - A Review of the State-of-the-Art*, The American Institute of Mining, Metallurgical, and Petroleum Engineers, Inc., New York, NY. pp. 539–562D.

Kawahata, K., 2006. *A New Algorithm to Solve Large Scale Mine Production Scheduling Problems by Using the Lagrangian Relaxation Method*. Ph.D. thesis, Colorado School of Mines.

Kim, Y., 1978. "Ultimate pit limit design methodologies using computer models - the state of the art." *Mining Engineering*, pp. 1454–1459.

Koenigsberg, E., 1982. "The optimum contours of an open pit mine: An application of dynamic programming." In T. Johnson and R. Barnes, editors, *17th APCOM*, Society of Mining Engineers of the American Institute of Mining, Metallurgical, and Petroleum Engineers, Inc., New York, NY, chapter 26. pp. 274–287.

Korobov, S., 1974. "Method for determining optimal open pit limits." Technical report, Dept of Mineral Engineering, Ecole Polytechnique de Montreal, Canada.

Kuchta, M., Newman, A., and Topal, E., 2003. "Production scheduling at LKAB's Kiruna mine using mixed integer programming." *Mining Engineering*, Vol. 55, No. 4, pp. 35–40.

Kumral, M. and Dowd, P., 2002. "Short-term mine production scheduling for industrial minerals using multi-objective simulated annealing." In S. Bandopadhyay, editor, *30th APCOM*, Society of Mining, Metallurgy, and Exploration, Inc, Littleton, CO. pp. 731–741.

Laurich, R., 1990. "Chapter 5 - planning and design of surface mines." Society for Mining, Metallurgy, and Exploration, Inc.

Lerchs, H. and Grossmann, I., 1965. "Optimum design of open-pit mines." *Transactions of the Canadian Institute of Mining and Metallurgy*, Vol. 68, pp. 17–24.

Menabde, M., Froyland, G., Stone, P., and Yeates, G., 2004. "Mining schedule optimization for conditionally simulated orebodies." In *Proceedings of Orebody Modeling and Strategic Mine Planning - Uncertainty and Risk Management*. Perth, Australia, pp. 353–357.

MineSight, 2006. "MineSight Economic Planner, version 1.0.0."

Nemhauser, G. and Wolsey, L., 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc.

Onur, A. and Dowd, P., 1993. "Open-pit optimization - part 2: Production scheduling and inclusion of roadways." *Transactions of the Institute of Mining & Metallurgy (Section A)*, Vol. 102, pp. A105–A113.

Osanloo, M., Gholamnejad, J., and Karimi, B., 2007. "Long-term open pit mine production planning: A review of models and algorithms." *International Journal of Mining, Reclamation and Environment*, Vol. 22, No. 1, pp. 1–33.

Ovanic, J. and Young, D., 1995. "Economic optimization of stope geometry using separable programming with special branch and bound techniques." In *Proceedings of the 3rd CAMI*. pp. 129–135.

Pana, M., 1965. "The simulation approach to open-pit design." In J. Dotson and W. Peters, editors, *Short Course and Symposium on Computers and Computer Applications in Mining and Exploration*, College of Mines, University of Arizona, Tuscon, Arizona. pp. ZZ–1 – ZZ–24.

Picard, J., 1976. "Maximal closure of a graph and applications to combinatorial problems." *Management Science*, Vol. 22, No. 11, pp. 1268–1272.

Ramazan, S., 2001. *Open Pit Mine Scheduling Based on Fundamental Tree Algorithm*. Dissertation, Colorado School of Mines, Golden, CO.

Ramazan, S., 2007. "The new fundamental tree algorithm for production scheduling of open pit mines." *European Journal of Operational Research*, Vol. 177, No. 2, pp. 1153–1166.

Ramazan, S. and Dimitrakopoulos, R., 2004a. "Recent applications of operations research and efficient MIP formulations in open pit mining." *Transactions of the Society for Mining, Metallurgy, and Exploration*, Vol. 316, pp. 73–78.

Ramazan, S. and Dimitrakopoulos, R., 2004b. "Traditional and new MIP models for production scheduling with in-situ grade variability." *International Journal of Surface Mining, Reclamation and Environment*, Vol. 18, No. 2, pp. 85–98.

Rardin, R., 1998. "Discrete optimization methods." In *Optimization in Operations Research*, Prentice Hall, Upper Saddle River, NJ, chapter 12. pp. 627–714.

Roman, R., 1974. "The role of time value of money in determining an open pit mining sequence and pit limits." In T. Johnson and D. Gentry, editors, *12th APCOM*, Colorado School of Mines, Golden, CO. pp. 72 – 85.

Sevim, H. and Lei, D., 1998. "The problem of production planning in open pit mines." *INFOR*, Vol. 33, No. 1/2, pp. 1–12.

Shenggui, Z. and Starfield, A., 1985. "Dynamic programming with color graphics smoothing for open-pit design on a personal computer." *International Journal of Mining Engineering*, Vol. 3, No. 1, pp. 27–34.

Tamatomi, J., Mogi, G., Akaike, A., and Yamaguchi, U., 1995. "Selective extractive dynamic cone algorithm for three-dimensional open pit designs." In *25th APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO. pp. 267–274.

Thomas, G., 1996. "Pit optimization and mine production scheduling - the way ahead." In R. Ramani, editor, *26th APCOM*, Society for Mining, Metallurgy, and Exploration, Inc, Littleton, CO, chapter 35. pp. 221–228.

Tolwinski, B., 1998. "Scheduling production for open pit mines." In *1998 APCOM*. pp. 19–23.

Tolwinski, B. and Golosinski, T., 1995. "Long term open pit scheduler." In *Proceedings of the International Symposium on Mine Planning and Equipment Selection*. pp. 256–270.

Tolwinski, B. and Underwood, R., 1992. "An algorithm to estimate the optimal evolution of an open pit mine." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 41. pp. 399–409.

Wang, Q. and Sevim, H., 1992. "Enhanced production planning in open pit mining through intelligent dynamic search." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 46. pp. 461–471.

Wang, Q. and Sevim, H., 1995. "Alternative to parameterization in finding a series of maximum-metal pits for production planning." *Mining Engineering*, pp. 178–182.

Wilke, F. and Wright, E., 1984. "Determining the optimal ultimate opencast pit design by dynamic programming." *Erzmetall*, Vol. 37, No. 3, pp. 138–144.

Wright, E., 1987. "The use of dynamic programming for open pit mine design: Some practical implications." *Mining Science and Technology*, Vol. 4, pp. 97–104.

Yegulalp, T., Arias, A., Muduli, P., and Xi, Y., 1993. "New developments in ultimate pit limit problem solution methods." *Transactions of the Society for Mining, Metallurgy, and Exploration*, Vol. 294, pp. 1853–1857.

Zhao, Y. and Kim, Y., 1992. "A new optimum pit limit design algorithm." In Y. Kim, editor, *23rd APCOM*, Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO, chapter 43. pp. 423–434.

APPENDIX A

The following tables present detailed results of employing our solution methodologies. Each table compares our techniques with the solution time if none of our techniques is used. The *monolith* column presents the results using the raw data with no modifications. The *ES & LS* column shows the results using the raw data with earliest and latest starts implemented. The *ES & LS & cuts* column depicts the raw data with earliest and latest starts and an appropriate level of cuts included. Lastly, the column labeled *Lagrangian Relaxation with ES & LS* presents the results from implementing the Lagrangian relaxation procedure on the data set with earliest and latest starts. Each table represents a separate data set, so we show each of the twelve data sets depicted in Table 5.1.

	monolith	ES & LS	ES & LS & cuts	Lagrangian Relaxation with ES & LS
# cuts	0	0	7,039	0
cut generation time (sec.)	0	0	62	0
# binary variables	6,360	5,025	5,025	5,025
# constraints	31,688	23,801	30,840	23,795
MIP simplex iterations	81,449	27,691	18,796	13,417
branch-and-bound nodes	360	120	60	0
computer time (sec.)	1,082	147	82	105
NPV ($\$10^6$)	19.0	19.0	19.1	18.8

Table A.1. Detailed Results for the *1,060* Data Set. This table shows the detailed results for the data set called *1,060*.

	monolith	ES & LS	ES & LS & cuts	Lagrangian Relaxation with ES & LS
# cuts	0	0	3,846	0
cut generation time (sec.)	0	0	46	0
# binary variables	11,880	9,975	9,975	9,975
# constraints	69,582	56,915	60,761	56,909
MIP simplex iterations	33,905	32,743	34,112	26,540
branch-and-bound nodes	30	40	40	0
computer time (sec.)	202	148	95	193
NPV ($\$10^6$)	17.5	17.6	17.6	17.2

Table A.2. Detailed Results for the *1,980* Data Set. This table shows the detailed results for the data set called *1,980*.

	monolith	ES & LS	ES & LS & cuts	Lagrangian Relaxation with ES & LS
# cuts	0	0	31,643	0
cut generation time (sec.)	0	0	284	0
# binary variables	17,280	14,367	14,367	14,367
# constraints	102,624	83,062	114,705	83,056
MIP simplex iterations	85,855	44,196	40,082	81,285
branch-and-bound nodes	50	80	60	0
computer time (sec.)	1,481	366	419	660
NPV ($\$10^6$)	15.5	15.6	15.6	15.2

Table A.3. Detailed Results for the *2,880* Data Set. This table shows the detailed results for the data set called *2,880*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,869	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	413,871	308,443	108,396	55,726
branch-and-bound nodes	100	570	280	0
computer time (sec.)	86,423	16,326	3,801	708
NPV ($\$10^6$)	9.1	9.2	9.2	9.1

Table A.4. Detailed Results for the *10,819* Data Set. This table shows the detailed results for the data set called *10,819*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,816	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	433,789	78,494	183,408	55,126
branch-and-bound nodes	131	120	440	0
computer time (sec.)	86,420	1,570	7,803	769
NPV ($\$10^6$)	9.3	9.3	9.3	9.1

Table A.5. Detailed Results for the *10,819A* Data Set. This table shows the detailed results for the data set called *10,819A*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,834	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	462,669	142,060	364,104	64,334
branch-and-bound nodes	80	360	220	40
computer time (sec.)	86,424	4,225	29,545	1,261
NPV ($\$10^6$)	9.1	9.2	9.2	9.1

Table A.6. Detailed Results for the *10,819B* Data Set. This table shows the detailed results for the data set called *10,819B*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,803	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	408,893	97,302	69,579	27,556
branch-and-bound nodes	139	320	200	0
computer time (sec.)	86,424	3,054	1,570	698
NPV ($\$10^6$)	9.1	9.2	9.2	9.0

Table A.7. Detailed Results for the *10,819C* Data Set. This table shows the detailed results for the data set called *10,819C*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,802	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	444,515	150,695	64,345	60,883
branch-and-bound nodes	95	420	168	25
computer time (sec.)	86,422	4,355	1,633	1,075
NPV ($\$10^6$)	9.2	9.2	9.2	9.0

Table A.8. Detailed Results for the *10,819D* Data Set. This table shows the detailed results for the data set called *10,819D*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	1,798	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	482,587	69,267	89,042	53,875
branch-and-bound nodes	100	217	200	0
computer time (sec.)	86,437	1,426	2,794	680
NPV ($\$10^6$)	9.0	9.1	9.2	9.0

Table A.9. Detailed Results for the *10,819E* Data Set. This table shows the detailed results for the data set called *10,819E*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	2,471	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	475,860	118,917	166,627	230,153
branch-and-bound nodes	95	320	320	516
computer time (sec.)	86,421	3,680	7,273	14,159
NPV ($\$10^6$)	N/A	9.3	9.3	9.1

Table A.10. Detailed Results for the *10,819F* Data Set. This table shows the detailed results for the data set called *10,819F*.

	monolith	ES	ES & cuts	Lagrangian Relaxation with ES
# cuts	0	0	19,223	0
cut generation time (sec.)	0	0	2,471	0
# binary variables	64,914	20,969	20,969	20,969
# constraints	395,885	117,038	136,261	117,032
MIP simplex iterations	446,595	56,542	197,578	81,726
branch-and-bound nodes	89	103	149	40
computer time (sec.)	86,431	838	14,121	3,687
NPV ($\$10^6$)	9.1	9.2	9.3	9.1

Table A.11. Detailed Results for the *10,819G* Data Set. This table shows the detailed results for the data set called *10,819G*.

	monolith	ES	ES & cuts
# cuts	0	0	2,010
cut generation time (sec.)	0	0	4
# binary variables	1,391	1,266	1,266
# linear variables	79,572	79,572	79,572
# constraints	31,929	31,097	32,950
MIP simplex iterations	441,279	365,085	356,262
branch-and-bound nodes	903	695	830
computer time (sec.)	11,476	9,719	8,696
NPV ($\$10^6$)	2.4	2.4	2.4

Table A.12. Detailed Results for the *Newmont* Data Set. This table shows the detailed results for the data set called *Newmont*.